深入理解LINUX网络内幕（影印版）

Understanding
LINUX
NETWORK INTERNALS

O'REILLY®
东南大学出版社

CHRISTIAN BENVENUTI 著

Understanding
LINUX NETWORK INTERNALS

深入理解LINUX网络内幕(影印版)

Christian Benvenuti

O'REILLY®
Beijing · Cambridge · Farnham · Köln · Paris · Sebastopol · Taipei · Tokyo
O'Reilly Media, Inc.授权东南大学出版社出版
东南大学出版社

图书在版编目（CIP）数据

深入理解Linux网络内幕 /（意）本文纳特（Benvenuti, C.）著. —影印本. —南京：东南大学出版社，2006.5

书名原文：Understanding Linux Network Internals

ISBN 7-5641-0367-1

Ⅰ.深 … Ⅱ.本 … Ⅲ.Linux操作系统－英文 Ⅳ.TP316.89

中国版本图书馆CIP数据核字（2006）第047702号

江苏省版权局著作权合同登记

图字：10-2006-131号

书　　名/ 深入理解Linux网络内幕（影印版）

书　　号/ ISBN 7-5641-0367-1

责任编辑/ 张烨

封面设计/ Karen Montgomey，张健

出版发行/ 东南大学出版社（press.seu.edu.cn）

地　　址/ 南京四牌楼2号（邮政编码210096）

印　　刷/ 扬中市印刷有限公司

开　　本/ 787毫米×980毫米　16开本　66.5印张

版　　次/ 2006年5月第1版　2006年5月第1次印刷

印　　数/ 0001-2000册

定　　价/ 98.00元（册）

O'Reilly Media, Inc. 介绍

O'Reilly Media, Inc. 是世界上在 UNIX、X、Internet 和其他开放系统图书领域具有领导地位的出版公司，同时是联机出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》（被纽约公共图书馆评为二十世纪最重要的 50 本书之一）到 GNN（最早的 Internet 门户和商业网站），再到 WebSite（第一个桌面PC的Web服务器软件），O'Reilly Media, Inc. 一直处于 Internet 发展的最前沿。

许多书店的反馈表明，O'Reilly Media, Inc. 是最稳定的计算机图书出版商 —— 每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media, Inc. 具有深厚的计算机专业背景，这使得 O'Reilly Media, Inc. 形成了一个非常不同于其他出版商的出版方针。O'Reilly Media, Inc. 所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly Media, Inc. 还有许多固定的作者群体 —— 他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly Media, Inc. 依靠他们及时地推出图书。因为 O'Reilly Media, Inc. 紧密地与计算机业界联系着，所以 O'Reilly Media, Inc. 知道市场上真正需要什么图书。

出版说明

随着计算机技术的成熟和广泛应用，人类正在步入一个技术迅猛发展的新时期。计算机技术的发展给人们的工业生产、商业活动和日常生活都带来了巨大的影响。然而，计算机领域的技术更新速度之快也是众所周知的，为了帮助国内技术人员在第一时间了解国外最新的技术，东南大学出版社和美国 O'Reilly Meida, Inc.达成协议，将陆续引进该公司的代表前沿技术或者在某专项领域享有盛名的著作，以影印版或者简体中文版的形式呈献给读者。其中，影印版书籍力求与国外图书“同步”出版，并且“原汁原味”展现给读者。

我们真诚地希望，所引进的书籍能对国内相关行业的技术人员、科研机构的研究人员和高校师生的学习和工作有所帮助，对国内计算机技术的发展有所促进。也衷心期望读者提出宝贵的意见和建议。

最新出版的一批影印版图书，包括：

- 《深入理解 Linux 内核 第三版》（影印版）
- 《Perl 最佳实践》（影印版）
- 《高级 Perl 编程 第二版》（影印版）
- 《Perl 语言入门 第四版》（影印版）
- 《深入浅出 HTML 与 CSS、XHTML》（影印版）
- 《UML 2.0 技术手册》（影印版）
- 《802.11 无线网络权威指南 第二版》（影印版）
- 《项目管理艺术》（影印版）
- 《.NET 组件开发 第二版》（影印版）
- 《ASP.NET 编程 第三版》（影印版）
- 《深入浅出 Ajax》（影印版）
- 《Ajax Hacks》（影印版）
- 《深入理解 Linux 网络内幕》（影印版）
- 《Web 设计技术手册 第三版》（影印版）
- 《软件预构艺术》（影印版）

Preface

Today more than ever before, networking is a hot topic. Any electronic gadget in its latest generation embeds some kind of networking capability. The Internet continues to broaden in its population and opportunities. It should not come as a surprise that a robust, freely available, and feature-rich operating system like Linux is well accepted by many producers of embedded devices. Its networking capabilities make it an optimal operating system for networking devices of any kind. The features it already has are well implemented, and new ones can be added easily. If you are a developer for embedded devices or a student who would like to experiment with Linux, this book will provide you with good fodder.

The performance of a pure software-based product that uses Linux cannot compete with commercial products that can count on the help of specialized hardware. This of course is not a criticism of software; it is a simple recognition of the consequence of the speed difference between dedicated hardware and general-purpose CPUs. However, Linux can definitely compete with low-end commercial products that are entirely software-based. Of course, simple extensions to the Linux kernel allow vendors to use Linux on hybrid systems as well (software and hardware); it is only a matter of writing the necessary device drivers.

Linux is also often used as the operating system of choice for the implementation of university projects and theses. Not all of them make it to the official kernel (not right away, at least). A few do, and others are simply made available online as patches to the official kernel. Isn't it a great satisfaction and reward to see your contribution to the Linux kernel being used by potentially millions of users? There is only one drawback: if your contribution is really appreciated, you may not be able to cope with the numerous emails of thanks or requests for help.

The momentum for Linux has been growing continually over the past years, and apparently it can only keep growing.

I first encountered Linux at the University of Bologna, where I was a grad student in computer science around 10 years ago. What a wonderful piece of software! I could

work on my image processing projects at home on an i286/486 computer without having to compete with other students for access to the few Sun stations available at the university labs.

Since then, my marriage to Linux has never seen a gray day. It has even started to displace my fond memories of the glorious C64 generation, when I was first introduced to programming with Assembly language and the various dialects of BASIC. Yes, I belong to the C64 generation, and to some extent I can compare the joy of my first programming experiences with the C64 to my first journeys into the Linux kernel.

When I was first introduced to the beautiful world of networking, I started playing with the tools available on Linux. I also had the fortune to work for a UNESCO center in Italy where I helped develop their networking courses, based entirely on Linux boxes. That gave me access to a good lab equipped with all sorts of network devices and documentation, plus plenty of Linux enthusiasts to learn from and to collaborate with.

Unfortunately for my own peace of mind (but fortunately, I hope, for the reader of this book who benefits from the results), I am the kind of person that likes to understand everything and takes very little for granted. So at UNESCO, I started looking into the kernel code. This not only proved to be a good way to burn in my knowledge, but it also gave me more confidence in making use of user-space configuration tools: whenever a configuration tool did not provide a specific option, I usually knew whether it would be possible to add it or whether it would have required significant changes to the kernel. This kind of study turns into a path without an end: you always want more.

After developing a few tools as extensions to the Linux kernel (some revision of versions 2.0 and 2.2), my love for operating systems and networking led me to the Silicon Valley (Cisco Systems). When you learn a language, be it a human language or a computer programming language, a rule emerges: the more languages you know, the easier it becomes to learn new ones. You can identify each one's strengths and weaknesses, see the reasons behind design compromises, etc. The same applies to operating systems.

When I noticed the lack of good documentation about the networking code of the Linux kernel and the availability of good books for other parts of the kernel, I decided to try filling in the gap—or at least part of it. I hope this book will give you the starting documentation that I would have loved to have had years ago.

I believe that this book, together with O'Reilly's other two kernel books (*Understanding the Linux Kernel* and *Linux Device Drivers*), represents a good starting point for anyone willing to learn more about the Linux kernel internals. They complement each other and, when they do not address a given feature, point the reader to external documentation sources (when available).

However, I still suggest you make some coffee, turn on the music, and spend some time on the source code trying to understand how a given feature is implemented. I believe the knowledge you build in this way lasts longer than that built in any other way. Shortcuts are good, but sometimes the long way has its advantages, too.

The Audience for This Book

This book can help those who already have some knowledge of networking and would like to see how the engine of the Internet—that is, the Internet Protocol (IP) and its friends—is implemented on a first-class operating system. However, there is a theoretical introduction for each topic, so newcomers will be able to get up to speed quickly, too. Complex topics are accompanied by enough examples to make them easier to follow.

Linux doesn't just support basic IP; it also has quite a few advanced features. More important, its implementation must be sophisticated enough to play nicely with other kernel features such as symmetric multiprocessing (SMP) and kernel preemption. This makes the networking code of the Linux kernel a very good gym in which to train and keep your networking knowledge in shape.

Moreover, if you are like me and want to learn everything, you will find enough details in this book to keep you satisfied for quite a while.

Background Information

Some knowledge of operating systems would help. The networking code, like any other component of the operating system, must follow both common sense and implicit rules for coexistence with the rest of the kernel, including proper use of locking; fair use of memory and CPU; and an eye toward modularity, code cleanliness, and good performance. Even though I occasionally spend time on those aspects, I refer you to the other two O'Reilly kernel books mentioned earlier for a deeper and detailed discussion on generic operating system services and design.

Some knowledge of networking, and especially IP, would also help. However, I think the theory overview that precedes each implementation description in this book is sufficient to make the book self-contained for both newcomers and experienced readers.

The theoretical description of the topics covered in the book does not require any programming experience. However, the descriptions of the associated implementations require an intermediate knowledge of the C language. Chapter 1 will go through a series of coding conventions and tricks that are often used in the code, which should help especially those with less experience with C and kernel programming.

Organization of the Material

Some aspects of networking code require as many as seven chapters, while for other aspects one chapter is sufficient. When the topic is complex or big enough to span different chapters, the part of the book devoted to that topic always starts with a concept chapter that covers the theory necessary to understand the implementation, which is described in another chapter. All of the reference and secondary material is usually located in one miscellaneous chapter at the end of the part. No matter how big the topic is, the same scheme is used to organize its presentation.

For each topic, the implementation description includes:

- The big picture, which shows where the described kernel component falls in the network stack.
- A brief description of the main data structures and a figure that shows how they relate to each other.
- A description of which other kernel features the component interfaces with—for example, by means of notification chains or data structure cross-references. The firewall is an example of such a kernel feature, given the numerous hooks it has all over the networking code.
- Extensive use of flow charts and figures to make it easier to go through the code and extract the logic from big and seemingly complex functions.

The reference material always includes:

- A detailed description of the most important data structures, field by field
- A table with a brief description of all functions, macros, and data structures, which you can use as a quick reference
- A list of the files mentioned in the chapter, with their location in the kernel source tree
- A description of the interface between the most common user-space tools used to configure the topic of the chapter and the kernel
- A description of any file in */proc* that is exported

The Linux kernel's networking code is not just a moving target, but a fast runner. The book does not cover all of the networking features. New ones are probably being added right now while you are reading. Many new features are driven by the needs of single users or organizations, or as university projects, but they find their way into the official kernel when they're considered useful for a large audience. Besides detailing the implementation of a subset of those features, I try to give you an idea of what the generic implementation of a feature might look like. This will help you greatly in understanding changes to the code and learning how new features are implemented. For example, given any feature, you need to take the following points into consideration:

- How do you design the data structures and the locking semantics?
- Is there a need for a user-space configuration tool? If so, is it going to interact with the kernel via an existing system call, an `ioctl` command, a */proc* file, or the Netlink socket?
- Is there any need for a new notification chain, and is there a need to register to an already existing chain?
- What is the relationship with the firewall?
- Is there any need for a cache, a garbage collection mechanism, statistics, etc.?

Here is the list of topics covered in the book:

Interface between user space and kernel
: In Chapter 3, you will get a brief overview of the mechanisms that networking configuration tools use to interact with their counterparts inside the kernel. It will not be a detailed discussion, but it will help you to understand certain parts of the kernel code.

System initialization
: Part II describes the initialization of key components of the networking code, and how network devices are registered and initialized.

Interface between device drivers and protocol handlers
: Part III offers a detailed description of how *ingress* (incoming or received) packets are handed by the device drivers to the upper-layer protocols, and vice versa.

Bridging
: Part IV describes transparent bridging and the Spanning Tree Protocol, the L2 (Layer two) counterpart of routing at L3 (Layer three).

Internet Protocol Version 4 (IPv4)
: Part V describes how packets are received, transmitted, forwarded, and delivered locally at the IPv4 layer.

Interface between IPv4 and the transport layer (L4) protocols
: Chapter 20 shows how IPv4 packets addressed to the local host are delivered to the transport layer (L4) protocols (TCP, UDP, etc.).

Internet Control Message Protocol (ICMP)
: Chapter 25 describes the implementation of ICMP, the only transport layer (L4) protocol covered in the book.

Neighboring protocols
: These find local network addresses, given their IP addresses. Part VI describes both the common infrastructure of the various protocols and the details of the ARP neighboring protocol used by IPv4.

Routing
: Part VII, the biggest one of the book, describes the routing cache and tables. Advanced features such as Policy Routing and Multipath are also covered.

What Is Not Covered

For lack of space, I had to select a subset of the Linux networking features to cover. No selection would make everyone happy, but I think I covered the core of the networking code, and with the knowledge you can gain with this book, you will find it easier to study on your own any other networking feature of the kernel.

In this book, I decided to focus on the networking code, from the interface between device drivers and the protocol handlers, up to the interface between the IPv4 and L4 protocols. Instead of covering all of the features with a compromise on quality, I preferred to keep quality as the first goal, and to select the subset of features that would represent the best start for a journey into the kernel networking implementation.

Here is a partial list of the features I could not cover for lack of space:

Internet Protocol Version 6 (IPv6)
: Even though I do not cover IPv6 in the book, the description of IPv4 can help you a lot in understanding the IPv6 implementation. The two protocols share naming conventions for functions and often for variables. Their interface to Netfilter is also similar.

IP Security protocol
: The kernel provides a generic infrastructure for cryptography along with a collection of both ciphers and digest algorithms. The first interface to the cryptographic layer was synchronous, but the latest improvements are adding an asynchronous interface to allow Linux to take advantage of hardware cards that can offload the work from the CPU.

  The protocols of the IPsec suite—Authentication Header (AH), Encapsulating-Security Payload (ESP), and IP Compression (IPcomp)—are implemented in the kernel and make use of the cryptographic layer.

IP multicast and IP multicast routing
: Multicast functionality was implemented to conform to versions 2 and 3 of the Internet Group Management Protocol (IGMP). Multicast routing support is also present, conforming to versions 1 and 2 of Protocol Independent Multicast (PIM).

Transport layer (L4) protocols
: Several L4 protocols are implemented in the Linux kernel. Besides the two well-known ones, UDP and TCP, Linux has the newer Stream Control Transmission Protocol (SCTP). A good description of the implementation of those protocols would require a new book of this size, all on its own.

Traffic Control
: This is the Quality of Service (QoS) layer of Linux, another interesting and powerful component of the kernel's networking code. Traffic control is implemented as a general infrastructure and as a collection of traffic classifiers and queuing disciplines. I briefly describe it and the interface it provides to the main transmission routine in Chapter 11. A great deal of documentation is available at *http://lartc.org*.

Netfilter

The firewall code infrastructure and its extensions (including the various NAT flavors) is not covered in the book, but I describe its interaction with most of the networking features I cover. At the Netfilter home page, *http://www.netfilter.org*, you can find some interesting documentation about its kernel internals.

Network filesystems

Several network filesystems are implemented in the kernel, among them NFS (versions 2, 3, and 4), SMB, Coda, and Andrew. You can read a detailed description of the Virtual File System layer in *Understanding the Linux Kernel*, and then delve into the source code to see how those network filesystems interface with it.

Virtual devices

The use of a dedicated virtual device underlies the implementation of networking features. Examples include 802.1Q, bonding, and the various tunneling protocols, such as IP-over-IP (IPIP) and Generalized Routing Encapsulation (GRE). Virtual devices need to follow the same guidelines as real devices and provide the same interface to other kernel components. In different chapters, where needed, I compare real and virtual device behaviors. The only virtual device that is described in detail is the bridge interface, which is covered in Part IV.

DECnet, IPX, AppleTalk, etc.

These have historical roots and are still in use, but are much less commonly used than IP. I left them out to give more space to topics that affect more users.

IP virtual server

This is another interesting piece of the networking code, described at *http://www.linuxvirtualserver.org/*. This feature can be used to build clusters of servers using different scheduling algorithms.

Simple Network Management Protocol (SNMP)

No chapter in this book is dedicated to SNMP, but for each feature, I give a description of all the counters and statistics kept by the kernel, the routines used to manipulate them, and the */proc* files used to export them, when available.

Frame Diverter

This feature allows the kernel to kidnap ingress frames not addressed to the local host. I will briefly mention it in Part III. Its home page is *http://diverter.sourceforge.net*.

Plenty of other network projects are available as separate patches to the kernel, and I can't list them all here. One that I find particularly fascinating and promising, especially in relation to the Linux routing code, is the highly configurable Click router, currently offered at *http://pdos.csail.mit.edu/click/*.

Because this is a book about the kernel, I do not cover user-space configuration tools. However, for each topic, I describe the interface between the most common user-space configuration tools and the kernel.

Conventions Used in This Book

The following is a list of the typographical conventions used in this book:

Italic
: Used for file and directory names, program and command names, command-line options, URLs, and new terms

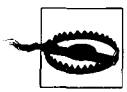
`Constant Width`
: Used in examples to show the contents of files or the output from commands, and in the text to indicate words that appear in C code or other literal strings

`Constant Width Italic`
: Used to indicate text within commands that the user replaces with an actual value

`Constant Width Bold`
: Used in examples to show commands or other text that should be typed literally by the user

Pay special attention to notes set apart from the text with the following icons:

This is a tip. It contains useful supplementary information about the topic at hand.



This is a warning. It helps you solve and avoid annoying problems.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. The code samples are covered by a dual BSD/GPL license.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Understanding Linux Network Internals*, by Christian Benvenuti. Copyright 2006 O'Reilly Media, Inc., 0-596-00255-6."

We'd Like to Hear from You

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international or local)
(707) 829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/understandlni/

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our web site at:

http://www.oreilly.com

Safari Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://safari.oreilly.com*.

Acknowledgments

This book would not have been possible without an interesting topic to talk about, and an audience. The interesting topic is Linux, this modern operating system that anyone has an opportunity to be part of, and the audience is the incredible number of users that often decide not only to take advantage of the good work of others, but also to contribute to its success by getting involved in its development. I have always loved sharing knowledge and passion for the things I like, and with this book, I have tried my best to add a lane or two to the highway that takes interested people into the wonderful world of the Linux kernel.

Of course, I did not do everything while lying in a hammock by the beach, with an ice cream in one hand and a mouse in the other. It took quite a lot of work to investigate the reasons behind some of the implementation choices. It is incredible how much information you can dig out of the development mailing lists, and how much people are willing to share their knowledge when you show genuine interest in their work.

For sure, this book would not be what it is without the great help and suggestions of my editor, Andy Oram. Due to the frequent changes that the networking code experiences, a few chapters had to undergo substantial updates during the writing of the book, but Andy understood this and helped me get to the finish line.

I also would like to thank all of those people that supported me in this effort, and Cisco Systems for giving me the flexibility I needed to work on this book.

A special thanks also goes to the technical reviewers for being able to review a book of this size in a short amount of time, still providing useful comments that allowed me to catch errors and improve the quality of the material. The book was reviewed by Jerry Cooperstein, Michael Boerner, and Paul Kinzelman (in alphabetical order, by first name). I also would like to thank Francois Tallet for reviewing Part IV and Andi Kleen for his feedback on Part V.

Table of Contents

Part II. System Initialization