HZ BOOKS
经 典 原 版 书 库
PRENTICE HALL PTR

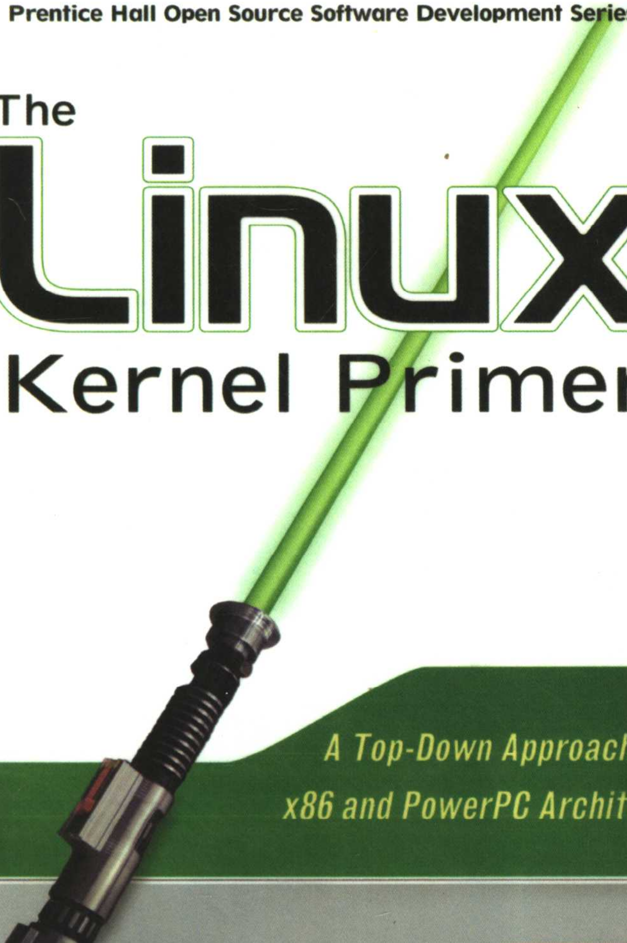
Linux内核编程必读

（英文版）

Prentice Hall Open Source Software Development Series
PRENTICE HALL PTR

The Linux®

Kernel Primer



A Top-Down Approach for x86 and PowerPC Architectures

（美） Claudia Salzberg Rodriguez
Gordon Fischer
Steven Smolski 著

机械工业出版社
China Machine Press

经 典 原 版 书 库

Linux内核编程必读

(英文版)

The Linux Kernel Primer

A Top-Down Approach for x86 and PowerPC Architectures

江苏工业学院图书馆
藏 书 章

(美) Claudia Salzberg Rodriguez
Gordon Fischer
Steven Smolski 著

机械工业出版社
China Machine Press

English reprint edition copyright © 2006 by Pearson Education Asia Limited and China Machine Press.

Original English language title: *The Linux Kernel Primer: A Top-Down Approach for x86 and PowerPC Architectures* (ISBN 0-13-118163-7) by Claudia Salzberg Rodriguez, Gordon Fischer, and Steven Smolski, Copyright © 2006 by Pearson Education, Inc.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Prentice Hall PTR.

For sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macau SAR).

本书英文影印版由Pearson Education Asia Ltd. 授权机械工业出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

仅限于中华人民共和国境内（不包括中国香港、澳门特别行政区和中国台湾地区）销售发行。

本书封面贴有Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

版权所有，侵权必究。
本书法律顾问　北京市展达律师事务所

本书版权登记号：图字：01-2006-3117

图书在版编目（CIP）数据

Linux内核编程必读（英文版）/（美）罗德里格斯（Rodriguez C. S.）等著. -北京：机械工业出版社，2006.7
（经典原版书库）
书名原文：The Linux Kernel Primer: A Top-Down Approach for x86 and PowerPC Architectures
ISBN 7-111-19345-8

Ⅰ.L…　Ⅱ.罗…　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062557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
北京诚信伟业印刷有限公司印刷 · 新华书店北京发行所发行
2006年7月第1版第1次印刷
170mm × 242mm · 40.25印张
定价：75.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

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

此为试读，需要完整PDF请访问：www.ertongbook.com

专家指导委员会

（按姓氏笔画顺序）

尤晋元　　王　珊　　冯博琴　　史忠植　　史美林

石教英　　吕　建　　孙玉芳　　吴世忠　　吴时霖

张立昂　　李伟琴　　李师贤　　李建中　　杨冬青

邵维忠　　陆丽娜　　陆鑫达　　陈向群　　周伯生

周克定　　周傲英　　孟小峰　　岳丽华　　范　明

郑国梁　　施伯乐　　钟玉琢　　唐世渭　　袁崇义

高传善　　梅　宏　　程　旭　　程时端　　谢希仁

裘宗燕　　戴　葵

To my parents, Pablo & Maria, por ser trigo, escudo, viento y bandera.
—Claudia Salzberg Rodriguez

To Lisa,

To Jan & Hart.

—Gordon Fischer

To my dear friend Wes, whose wisdom and friendship I will cherish forever.
—Steven Smolski

Foreword

Here there be dragons. Medieval mapmakers wrote that about unknown or dangerous places, and that is likely the feeling you get the first time you type:

```
cd /usr/src/linux ; ls
```

"Where do I start?" you wonder. "What exactly am I looking at? How does it all hang together and actually work?"

Modern, full-featured operating systems are big and complex. The number of subsystems is large, and their interactions are many and often subtle. And while it's great that you *have* the Linux kernel source code (more about that in a moment), knowing where to start, what to look at, and in what order, is far from self-evident.

That is the purpose of this book. Step by step, you will learn about the different kernel components, how they work, and how they relate to each other. The authors are intimately familiar with the kernel, and this knowledge shows through; by the end of the book, you and the kernel will at least be good friends, with the prospect of a deeper relationship ahead of you.

The Linux kernel is "Free" (as in freedom) Software. In *The Free Software Definition,*[1] Richard Stallman defines the freedoms that make software Free (with a capital F). Freedom 0 is the freedom to run the software. This is the most fundamental freedom. But immediately after that is Freedom 1, the freedom to study how a program works. This freedom is often overlooked. However, it is very important, because one of the best ways to learn how to do something is by watching other people do it. In the software world, that means reading other peoples' programs and

---

[1] `http://www.gnu.org/philosophy/free-sw.html`

seeing what they did well as well as what they did poorly. The freedoms of the GPL are, at least in my opinion, one of the most fundamental reasons that GNU/Linux systems have become such an important force in modern computing. Those freedoms benefit you every moment you use your GNU/Linux system, and it's a good idea to stop and think about that every once in awhile.

With this book, we take advantage of Freedom 1 to give you the opportunity to study the Linux kernel source code in depth. You will see things that are done well, and other things that are done, shall we say, *less well.* But because of Freedom 1, *you will see it all,* and you will be able to learn from it.

And that brings me to the *Prentice Hall Open Source Software Development Series,* of which this book is one of the first members. The idea for the series developed from the principle that reading programs is one of the best ways to learn. Today, the world is blessed with an abundance of Free and Open Source software—whose source code is just waiting (maybe even eager!) to be read, understood, and appreciated. The aim of the series is to be your guide up the software development learning curve, so to speak, and to help you learn by showing you as much real code as possible.

I sincerely hope that you will enjoy this book and learn a lot. I also hope that you will be inspired to carve out your own niche in the Free Software and Open Source worlds, which is definitely the most enjoyable way to participate in them.

Have fun!

Arnold Robbins
Series Editor

Acknowledgments

We would like to thank the many people without whom this book would not have been possible.

Claudia Salzberg Rodriguez: I would like to note that it is oftentimes difficult, when faced with a finite amount of space in which to acknowledge people, to distinguish the top contributors to your current and well-defined accomplishment from the mass of humanity which has, in countless and innumerable ways, contributed to you being capable of this accomplishment. That being said, I would like to thank all the contributors to the Linux kernel for all the hard work and dedication that has gone into developing this operating system into what it has become—for love of the game. My deepest appreciation goes out to the many key teachers and mentors along the way for awakening and fostering the insatiable curiosity for how things work and for teaching me how to learn. I would also like to thank my family for their constant love, support, and for maintaining their enthusiasm well past the point where mine was exhausted. Finally, I wish to thank Jose Raul, for graciously handling the demands on my time and for consistently finding the way to rekindle inspiration that insisted on giving out.

Gordon Fischer: I would like to thank all the programmers who patiently explained to me the intricacies of the Linux kernel when I was but a n00b. I would also like to thank Grady and Underworld for providing excellent coding music.

We would all like to thank our superb editor, Mark L. Taub, for knowing what was necessary to make the book better every step of the way and for leading us in that direction. Thank you for being constantly and simultaneously reasonable, understanding, demanding, and vastly accessible throughout the writing of this book.

We would also like to thank Jim Markham and Erica Jamison. Jim Markham we thank for his early editorial comments that served us so well throughout the rest of the writing of the manuscript. Erica Jamison we thank for providing us with editorial feedback during the last version of the manuscript.

Our appreciation flows out to our reviewers who spent so many hours reading and making suggestions that made the book better. Thank you for your keen eyes and insightful comments; your suggestions and comments were invaluable. The reviewers are (in alphabetical order) Alessio Gaspar, Mel Gorman, Benjamin Herrenschmidt, Ron McCarty, Chet Ramey, Eric Raymond, Arnold Robbins, and Peter Salus.

We would like to thank Kayla Dugger for driving us through the copyediting and proofreading process with unwavering good cheer, and Ginny Bess for her hawk-eyed copyedit. A special thanks goes to the army of people behind the scenes of the copyediting, proofreading, layout, marketing, and printing who we did not get to meet personally for making this book possible.

此为试读，需要完整PDF请访问：www.ertongbook.com

About the Authors

Claudia Salzberg Rodriguez works in IBM's Linux Technology Center, developing the kernel and associated programming tools. A Linux systems programmer for over five years, she has worked with Linux for Intel and PPC on platforms ranging from embedded to high-performance systems.

Gordon Fischer has written Linux and UNIX device drivers for many low-level devices, and has used Linux kernels in diverse enterprise settings across both Intel and PPC platforms.

Steve Smolski has been in the semiconductor business for 26 years. He has worked in the manufacturing, testing, and development of memory, processors, and ASICS; has written applications and drivers for Linux, AIX, and Windows; and has embedded operating systems.

Preface

Technology in general and computers in specific have a magical allure that seems to consume those who would approach them. Developments in technology push established boundaries and force the re-evaluation of troublesome concepts previously laid to rest. The Linux operating system has been a large contributor to a torrent of notable shifts in industry and the way business is done. By its adoption of the GNU Public License and its interactions with GNU software, it has served as a cornerstone to the various debates that surround open source, free software, and the concept of the development community. Linux is an extremely successful example of how powerful an open source operating system can be, and how the magic of its underpinnings can hold programmers from all corners of the world spellbound.

The use of Linux is something that is increasingly accessible to most computer users. With multiple distributions, community support, and industry backing, the use of Linux has also found safe harbor in universities, industrial applications, and the homes of millions of users.

Increased need in support and for new functionality follow at the heels of this upsurge in use. In turn, more and more programmers are finding themselves interested in the internals of the Linux kernel as the number of architectures and devices that demand support are added to the already vast (and rapidly growing) arsenal.

The porting of the Linux kernel to the Power architecture has contributed to the operating system's blossoming among high-end servers and embedded systems. The need for understanding how Linux runs on the Power architecture

has grown, with companies now purchasing PowerPC-based systems intended to run Linux.

Intended Audience

This book is intended for the budding and veteran systems programmer, the Linux enthusiast, and the application programmer eager to have a better understanding of what makes his programs work the way they do. Anyone who has knowledge of C, familiarity with basic Linux user fundamentals, and wants to know how Linux works should find this book provides him with the basic concepts necessary to build this understanding—it is intended to be a primer for understanding how the Linux kernel works.

Whether your experience with Linux has been logging in and writing small programs to run on Linux, or you are an established systems programmer seeking to understand particularities of one of the subsystems, this book provides you with the information you are looking for.

Organization of Material

This book is divided into three parts, each of which provides the reader with knowledge necessary to succeed in the study of Linux internals.

Part I provides the necessary tools and understanding to tackle the exploration of the kernel internals:

Chapter 1, "Overview," provides a history of Linux and UNIX, a listing of the many distributions, and a short overview of the various kernel subsystems from a user space perspective.

Chapter 2, "Exploration Toolkit," provides a description of the data structures and language usage commonly found throughout the Linux kernel, an introduction to assembly for x86 and PowerPC architectures, and a summary of tools and utilities used to get the information needed to understand kernel internals.

Part II introduces the reader to the basic concepts in each kernel subsystem and to trace the code that executes the subsystem functionality:

Chapter 3, "Processes: The Principal Model of Execution," covers the implementation of the process model. We explain how processes come to be and discuss the flow of control of a user space process into kernel space and back. We also discuss how processes are implemented in the kernel and discuss all data structures associated with process execution. This chapter also covers interrupts and exceptions, how these hardware mechanisms occur in each of the architectures, and how they interact with the Linux kernel.

Chapter 4, "Memory Management," describes how the Linux kernel tracks and manages available memory among various user space processes and the kernel. This chapter describes the way in which the kernel categorizes memory and how it decides to allocate and deallocate memory. It also describes in detail the mechanism of the page fault and how it is executed in the hardware.

Chapter 5, "Input/Output," describes how the processor interacts with other devices, and how the kernel interfaces and controls these interactions. This chapter also covers various kinds of devices and their implementation in the kernel.

Chapter 6, "Filesystems," provides an overview of how files and directories are implemented in the kernel. This chapter introduces the virtual filesystem, the layer of abstraction used to support multiple filesystems. This chapter also traces the execution of file-related operations such as open and close.

Chapter 7, "Scheduling and Kernel Synchronization," describes the operation of the scheduler, which allows multiple processes to run as though they are the only process in the system. This chapter covers in detail how the kernel selects which task to execute and how it interfaces with the hardware to switch from one process to another. This chapter also describes what kernel preemption is and how it is executed. Finally, it describes how the system clock works and its use by the kernel to keep time.

Chapter 8, "Booting the Kernel," describes what happens from Power On to Power Off. It traces how the various processors handle the loading of the kernel, including a description of BIOS, Open Firmware, and bootloaders. This chapter then goes through the linear order in kernel bringup and initialization, covering all the subsystems discussed in previous chapters.

Part III deals with a more hands-on approach to building and interacting with the Linux kernel:

Chapter 9, "Building the Linux Kernel," covers the toolchain necessary to build the kernel and the format of the object files executed. It also describes in detail how the Kernel Source Build system operates and how to add configuration options into the kernel build system.

Chapter 10, "Adding Your Code to the Kernel," describes the operation of `/dev/random`, which is seen in all Linux systems. As it traces the device, the chapter touches on previously described concepts from a more practical perspective. It then covers how to implement your own device in the kernel.

Our Approach

This book introduces the reader to the concepts necessary to understand the kernel. We follow a top-down approach in the following two ways:

First, we associate the kernel workings with the execution of user space operations the reader may be more familiar with and strive to explain the kernel workings in association with this. When possible, we begin with a user space example and trace the execution of the code down into the kernel. It is not always possible to follow this tracing straight down since the subsystem data types and substructures need to be introduced before the explanation of how it works can take place. In these cases, we tie in explanations of the kernel subsystem with specific examples of how it relates to a user space program. The intent is twofold: to highlight the layering seen in the kernel as it interfaces with user space on one side and the hardware on the other, and to explain workings of the subsystem by tracing the code and following the order of events as they occur. We believe this will help the reader get a sense of how the kernel workings fit in with what he knows, and will provide him with a framed reference for how a particular functionality associates to the rest of the operating system.

Second, we use the top-down perspective to view the data structures central to the operation of the subsystem and see how they relate to the execution of the system's management. We strive to delineate structures central to the subsystem operation and to keep focus on them as we follow the operation of the subsystem.

Conventions

Throughout this book, you will see listings of the source code. The top-right corner will hold the location of the source file with respect to the root of the source code tree. The listings are shown in `this font`. Line numbers are provided for the code commentary that usually follows. As we explain the kernel subsystem and how it works, we will continually refer to the source code and explain it.

Command-line options, function names, function output, and variable names are distinguished by `this font`.

Bold type is used whenever a new concept is introduced.

此为试读，需要完整PDF请访问：www.ertongbook.com