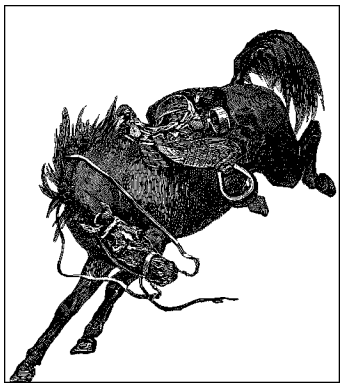
PREFACE



This is, on the surface, a book about writing device drivers for the Linux system. That is a worthy goal, of course; the flow of new hardware products is not likely to slow down anytime soon, and somebody is going to have to make all those new gadgets work with Linux. But this book is also about how the Linux kernel works and how to adapt its workings to your needs or interests. Linux is an open system; with this book, we hope, it will be more open and accessible to a larger community of developers.

Much has changed with Linux since the first edition of this book came out. Linux now runs on many more processors and supports a much wider variety of hardware. Many of the internal programming interfaces have changed significantly. Thus, the second edition. This book covers the 2.4 kernel, with all of the new features that it provides, while still giving a look backward to earlier releases for those who need to support them.

We hope you'll enjoy reading this book as much as we have enjoyed writing it.

Alessandro's Introduction

As an electronic engineer and a do-it-yourself kind of person, I have always enjoyed using the computer to control external hardware. Ever since the days of my father's Apple IIe, I have been looking for another platform where I could connect my custom circuitry and write my own driver software. Unfortunately, the PC of the 1980s wasn't powerful enough, at either the software or the hardware level: the internal design of the PC is much worse than that of the Apple II, and the available documentation has long been unsatisfying. But then Linux appeared, and I decided to give it a try by buying an expensive 386 motherboard and no proprietary software at all.

At the time, I was using Unix systems at the university and was greatly excited by the smart operating system, in particular when supplemented by the even smarter utilities that the GNU project donates to the user base. Running the Linux kernel on my own PC motherboard has always been an interesting experience, and I could even write my own device drivers and play with the soldering iron once again. I continue to tell people, "When I grow up, I wanna be a hacker," and GNU/Linux is the perfect platform for such dreams. That said, I don't know if I will ever grow up.

As Linux matures, more and more people get interested in writing drivers for custom circuitry and for commercial devices. As Linus Torvalds noted, "We're back to the times when men were men and wrote their own device drivers."

Back in 1996, I was hacking with my own toy device drivers that let me play with some loaned, donated, or even home-built hardware. I already had contributed a few pages to the *Kernel Hacker's Guide*, by Michael Johnson, and began writing kernel-related articles for *Linux Journal*, the magazine Michael founded and directed. Michael put me in touch with Andy Oram at O'Reilly; he expressed an interest in having me write a whole book about device drivers, and I accepted this task, which kept me pretty busy for quite a lot of time.

In 1999 it was clear I couldn't find the energy to update the book by myself: my family had grown and I had enough programming work to keep busy producing exclusively GPL'd software. Besides, the kernel had grown bigger and supported more diverse platforms than it used to, and the API had turned more broad and more mature. That's when Jonathan offered to help: he had just the right skills and enthusiasm to start the update and to force me to stay on track with the schedule—which slipped quite a lot anyway. He's been an invaluable mate in the process, which he pushed forward with good skills and dedication, definitely more than I could put in. I really enjoyed working with him, both on a technical and personal level.

Jon's Introduction

I first started actively playing with Linux early in 1994, when I convinced my employer to buy me a laptop from a company called, then, Fintronic Systems. Having been a Unix user since the beginning of the 1980s, and having played around in the source since about then, I was immediately hooked. Even in 1994, Linux was a highly capable system, and the first truly free system that I had ever been able to work with. I lost almost all my interest in working with proprietary systems at that point.

I didn't ever really plan to get into writing about Linux, though. Instead, when I started talking with O'Reilly about helping with the second edition of this book, I had recently quit my job of 18 years to start a Linux consulting company. As a way

of attracting attention to ourselves, we launched a Linux news site, Linux Weekly News (*http://lwn.net*), which, among other things, covered kernel development. As Linux exploded in popularity, the web site did too, and the consulting business was eventually forgotten.

But my first interest has always been systems programming. In the early days, that interest took the form of "fixing" the original BSD Unix paging code (which has to have been a horrible hack job) or making recalcitrant tape drives work on a VAX/VMS system (where source was available, if you didn't mind the fact that it was in assembly and Bliss, and came on microfiche only). As time passed, I got to hack drivers on systems with names like Alliant, Ardent, and Sun, before moving into tasks such as deploying Linux as a real-time radar data collection system or, in the process of writing this book, fixing the I/O request queue locking in the Linux floppy driver.

So I welcomed the opportunity to work on this book for several reasons. As much as anything, it was a chance to get deeply into the code and to help others with a similar goal. Linux has always been intended to be fun as well as useful, and playing around with the kernel is one of the most fun parts of all—at least, for those with a certain warped sense of fun. Working with Alessandro has been a joy, and I must thank him for trusting me to hack on his excellent text, being patient with me as I came up to speed and as I broke things, and for that jet-lagged bicycle tour of Pavia. Writing this book has been a great time.

Audience of This Book

On the technical side, this text should offer a hands-on approach to understanding the kernel internals and some of the design choices made by the Linux developers. Although the main, official target of the book is teaching how to write device drivers, the material should give an interesting overview of the kernel implementation as well.

Although real hackers can find all the necessary information in the official kernel sources, usually a written text can be helpful in developing programming skills. The text you are approaching is the result of hours of patient grepping through the kernel sources, and we hope the final result is worth the effort it took.

This book should be an interesting source of information both for people who want to experiment with their computer and for technical programmers who face the need to deal with the inner levels of a Linux box. Note that "a Linux box" is a wider concept than "a PC running Linux," as many platforms are supported by our operating system, and kernel programming is by no means bound to a specific platform. We hope this book will be useful as a starting point for people who want to become kernel hackers but don't know where to start.

The Linux enthusiast should find in this book enough food for her mind to start playing with the code base and should be able to join the group of developers that is continuously working on new capabilities and performance enhancements. This book does not cover the Linux kernel in its entirety, of course, but Linux device driver authors need to know how to work with many of the kernel's subsystems. It thus makes a good introduction to kernel programming in general. Linux is still a work in progress, and there's always a place for new programmers to jump into the game.

If, on the other hand, you are just trying to write a device driver for your own device, and you don't want to muck with the kernel internals, the text should be modularized enough to fit your needs as well. If you don't want to go deep into the details, you can just skip the most technical sections and stick to the standard API used by device drivers to seamlessly integrate with the rest of the kernel.

The main target of this book is writing kernel modules for version 2.4 of the Linux kernel. A *module* is object code that can be loaded at runtime to add new functionality to a running kernel. Wherever possible, however, our sample code also runs on versions 2.2 and 2.0 of the kernel, and we point out where things have changed along the way.

Organization of the Material

The book introduces its topics in ascending order of complexity and is divided into two parts. The first part (Chapters 1 to 10) begins with the proper setup of kernel modules and goes on to describe the various aspects of programming that you'll need in order to write a full-featured driver for a char-oriented device. Every chapter covers a distinct problem and includes a "symbol table" at the end, which can be used as a reference during actual development.

Throughout the first part of the book, the organization of the material moves roughly from the software-oriented concepts to the hardware-related ones. This organization is meant to allow you to test the software on your own computer as far as possible without the need to plug external hardware into the machine. Every chapter includes source code and points to sample drivers that you can run on any Linux computer. In Chapter 8 and Chapter 9, however, we'll ask you to connect an inch of wire to the parallel port in order to test out hardware handling, but this requirement should be manageable by everyone.

The second half of the book describes block drivers and network interfaces and goes deeper into more advanced topics. Many driver authors will not need this material, but we encourage you to go on reading anyway. Much of the material found there is interesting as a view into how the Linux kernel works, even if you do not need it for a specific project.

Background Information

In order to be able to use this book, you need to be confident with C programming. A little Unix expertise is needed as well, as we often refer to Unix commands and pipelines.

At the hardware level, no previous expertise is required to understand the material in this book, as long as the general concepts are clear in advance. The text isn't based on specific PC hardware, and we provide all the needed information when we do refer to specific hardware.

Several free software tools are needed to build the kernel, and you often need specific versions of these tools. Those that are too old can lack needed features, while those that are too new can occasionally generate broken kernels. Usually, the tools provided with any current distribution will work just fine. Tool version requirements vary from one kernel to the next; consult *Documentation/Changes* in the source tree of the kernel you are using for exact requirements.

Sources of Further Information

Most of the information we provide in this book is extracted directly from the kernel sources and related documentation. In particular, pay attention to the *Documentation* directory that is found in the kernel source tree. There is a wealth of useful information there, including documentation of an increasing part of the kernel API (in the *DocBook* subdirectory).

There are a few interesting books out there that extensively cover related topics; they are listed in the bibliography.

There is much useful information available on the Internet; the following is a sampling. Internet sites, of course, tend to be highly volatile while printed books are hard to update. Thus, this list should be regarded as being somewhat out of date.

http://www.kernel.org
ftp://ftp.kernel.org
: This site is the home of Linux kernel development. You'll find the latest kernel release and related information. Note that the FTP site is mirrored throughout the world, so you'll most likely find a mirror near you.

http://www.linuxdoc.org
: The Linux Documentation Project carries a lot of interesting documents called "HOWTOs"; some of them are pretty technical and cover kernel-related topics.

http://www.linux-mag.com/depts/gear.html
The "Gearheads only" section from *Linux Magazine* often runs kernel-oriented articles from well-known developers.

http://www.linux.it/kerneldocs
This page contains many kernel-oriented magazine articles written by Alessandro.

http://lwn.net
At the risk of seeming self-serving, we'll point out this news site (edited by one of your authors) which, among other things, offers regular kernel development coverage.

http://kt.zork.net
Kernel Traffic is a popular site that provides weekly summaries of discussions on the Linux kernel development mailing list.

http://www.atnf.csiro.au/~rgooch/linux/docs/kernel-newsflash.html
The Kernel Newsflash site is a clearinghouse for late-breaking kernel news. In particular, it concentrates on problems and incompatibilities in current kernel releases; thus, it can be a good resource for people trying to figure out why the latest development kernel broke their drivers.

http://www.kernelnotes.org
Kernel Notes is a classic site with information on kernel releases, unofficial patches, and more.

http://www.kernelnewbies.org
This site is oriented toward new kernel developers. There is beginning information, an FAQ, and an associated IRC channel for those looking for immediate assistance.

http://lksr.org
The Linux Kernel Source Reference is a web interface to a CVS archive containing an incredible array of historical kernel releases. It can be especially useful for finding out just when a particular change occurred.

http://www.linux-mm.org
This page is oriented toward Linux memory management development. It contains a fair amount of useful information and an exhaustive list of kernel-oriented web links.

http://www.conecta.it/linux
This Italian site is one of the places where a Linux enthusiast keeps updated information about all the ongoing projects involving Linux. Maybe you already know an interesting site with HTTP links about Linux development; if not, this one is a good starting point.

Online Version and License

The authors have chosen to make this book freely available under the GNU Free Documentation License, version 1.1.

Full license
http://www.oreilly.com/catalog/linuxdrive2/chapter/licenseinfo.html;

HTML
http://www.oreilly.com/catalog/linuxdrive2/chapter/book;

DocBook
http://www.oreilly.com/catalog/linuxdrive2/chapter/bookindex.xml;

PDF
http://www.oreilly.com/catalog/linuxdrive2/chapter/bookindexpdf.html.

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 Italic`	Used to indicate variable options, keywords, or text that the user is to replace with an actual value
`Constant 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.

We'd Like to Hear from You

We have tested and verified the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing to:

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international/local)
(707) 829-0104 (fax)

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

http://www.oreilly.com/catalog/linuxdrive2

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

bookquestions@oreilly.com

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

http://www.oreilly.com

Acknowledgments

This book, of course, was not written in a vacuum; we would like to thank the many people who have helped to make it possible.

I (Alessandro) would like to thank the people that made this work possible. First of all, the incredible patience of Federica, who went as far as letting me review the first edition during our honeymoon, with a laptop in the tent. Giorgio and Giulia have only been involved in the second edition of the book, and helped me stay in touch with reality by eating pages, pulling wires, and crying for due attention. I must also thank all four grandparents, who came to the rescue when the deadlines were tight and took over my fatherly duties for whole days, letting me concentrate on code and coffee. I still owe a big thanks to Michael Johnson, who made me enter the world of writing. Even though this was several years ago, he's still the one that made the wheel spin; earlier, I had left the university to avoid writing articles instead of software. Being an independent consultant, I have no employer that kindly allowed me to work on the book; on the other hand, I owe due acknowledgment to Francesco Magenta and Rodolfo Giometti, who are helping me as "dependent consultants." Finally, I want to acknowledge the free-software authors who actually taught me how to program without even knowing me; this

includes both kernel and user-space authors I enjoyed reading, but they are too many to list.

I (Jon) am greatly indebted to many people; first and foremost I wish to thank my wife, Laura, who put up with the great time demands of writing a book while simultaneously trying to make a "dotcom" business work. My children, Michele and Giulia, have been a constant source of joy and inspiration. Numerous people on the linux-kernel list showed great patience in answering my questions and setting me straight on things. My colleagues at LWN.net have been most patient with my distraction, and our readers' support of the LWN kernel page has been outstanding. This edition probably would not have happened without the presence of Boulder's local community radio station (appropriately named KGNU), which plays amazing music, and the Lake Eldora ski lodge, which allowed me to camp out all day with a laptop during my kids' ski lessons and served good coffee. I owe gratitude to Evi Nemeth for first letting me play around in the early BSD source on her VAX, to William Waite for *really* teaching me to program, and to Rit Carbone of the National Center for Atmospheric Research (NCAR), who got me started on a long career where I learned almost everything else.

We both wish to thank our editor, Andy Oram; this book is a vastly better product as a result of his efforts. And obviously we owe a lot to the smart people who pushed the free-software idea and still keep it running (that's mainly Richard Stallman, but he's definitely not alone).

We have also been helped at the hardware level; we couldn't study so many platforms without external help. We thank Intel for loaning an early IA-64 system, and Rebel.com for donating a Netwinder (their ARM-based tiny computer). Prosa Labs, the former Linuxcare-Italia, loaned a pretty fat PowerPC system; NEC Electronics donated their interesting development system for the VR4181 processor—that's a palmtop where we could put a GNU/Linux system on flash memory. Sun-Italia loaned both a SPARC and a SPARC64 system. All of those companies and those systems helped keep Alessandro busy in debugging portability issues and forced him to get one more room to fit his zoo of disparate silicon beasts.

The first edition was technically reviewed by Alan Cox, Greg Hankins, Hans Lermen, Heiko Eissfeldt, and Miguel de Icaza (in alphabetic order by first name). The technical reviewers for the second edition were Allan B. Cruse, Christian Morgner, Jake Edge, Jeff Garzik, Jens Axboe, Jerry Cooperstein, Jerome Peter Lynch, Michael Kerrisk, Paul Kinzelman, and Raph Levien. Together, these people have put a vast amount of effort into finding problems and pointing out possible improvements to our writing.

Last but certainly not least, we thank the Linux developers for their relentless work. This includes both the kernel programmers and the user-space people, who often get forgotten. In this book we chose never to call them by name in order to avoid being unfair to someone we might forget. We sometimes made an exception to this rule and called Linus by name; we hope he doesn't mind, though.