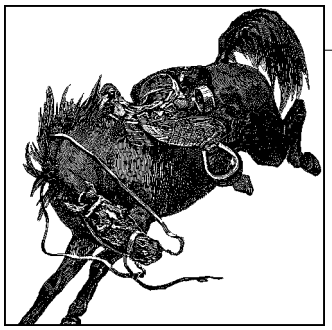

Preface

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 is more open and accessible to a larger community of developers.

This is the third edition of *Linux Device Drivers*. The kernel has changed greatly since this book was first published, and we have tried to evolve the text to match. This edition covers the 2.6.10 kernel as completely as we are able. We have, this time around, elected to omit the discussion of backward compatibility with previous kernel versions. The changes from 2.4 are simply too large, and the 2.4 interface remains well documented in the (freely available) second edition.

This edition contains quite a bit of new material relevant to the 2.6 kernel. The discussion of locking and concurrency has been expanded and moved into its own chapter. The Linux device model, which is new in 2.6, is covered in detail. There are new chapters on the USB bus and the serial driver subsystem; the chapter on PCI has also been enhanced. While the organization of the rest of the book resembles that of the earlier editions, every chapter has been thoroughly updated.

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

Jon's Introduction

The publication of this edition coincides with my twelth year of working with Linux and, shockingly, my twenty-fifth year in the computing field. Computing seemed like a fast-moving field back in 1980, but things have sped up a lot since then. Keeping *Linux Device Drivers* up to date is increasingly a challenge; the Linux kernel hackers continue to improve their code, and they have little patience for documentation that fails to keep up.

Linux continues to succeed in the market and, more importantly, in the hearts and minds of developers worldwide. The success of Linux is clearly a testament to its technical quality and to the numerous benefits of free software in general. But the true key to its success, in my opinion, lies in the fact that it has brought the fun back to computing. With Linux, anybody can get their hands into the system and play in a sandbox where contributions from any direction are welcome, but where technical excellence is valued above all else. Linux not only provides us with a top-quality operating system; it gives us the opportunity to be part of its future development and to have fun while we're at it.

In my 25 years in the field, I have had many interesting opportunities, from programming the first Cray computers (in Fortran, on punch cards) to seeing the minicomputer and Unix workstation waves, through to the current, microprocessor-dominated era. Never, though, have I seen the field more full of life, opportunity, and fun. Never have we had such control over our own tools and their evolution. Linux, and free software in general, is clearly the driving force behind those changes.

My hope is that this edition helps to bring that fun and opportunity to a new set of Linux developers. Whether your interests are in the kernel or in user space, I hope you find this book to be a useful and interesting guide to just how the kernel works with the hardware. I hope it helps and inspires you to fire up your editor and to make our shared, free operating system even better. Linux has come a long way, but it is also just beginning; it will be more than interesting to watch—and participate in—what happens from here.

Alessandro's Introduction

I've always enjoyed computers because they can talk to external hardware. So, after soldering my devices for the Apple II and the ZX Spectrum, backed with the Unix and free software expertise the university gave me, I could escape the DOS trap by installing GNU/Linux on a fresh new 386 and by turning on the soldering iron once again.

Back then, the community was a small one, and there wasn't much documentation about writing drivers around, so I started writing for Linux Journal. That's how things started: when I later discovered I didn't like writing papers, I left the univeristy and found myself with an O'Reilly contract in my hands.

That was in 1996. Ages ago.

The computing world is different now: free software looks like a viable solution, both technically and politically, but there's a lot of work to do in both realms. I hope this book furthers two aims: spreading technical knowledge and raising awareness about the need to spread knowledge. That's why, after the first edition proved interesting to the public, the two authors of the second edition switched to a free license,

supported by our editor and our publisher. I'm betting this is the right approach to information, and it's great to team up with other people sharing this vision.

I'm excited by what I witness in the embedded arena, and I hope this text helps by doing more; but ideas are moving fast these days, and it's already time to plan for the fourth edition, and look for a fourth author to help.

Greg's Introduction

It seems like a long time ago that I picked up the first edition of this *Linux Device Drivers* book in order to figure out how to write a real Linux driver. That first edition was a great guide to helping me understand the internals of this operating system that I had already been using for a number of years but whose kernel had never taken the time to look into. With the knowledge gained from that book, and by reading other programmers' code already present in the kernel, my first horribly buggy, broken, and very SMP-unsafe driver was accepted by the kernel community into the main kernel tree. Despite receiving my first bug report five minutes later, I was hooked on wanting to do as much as I could to make this operating system the best it could possibly be.

I am honored that I've had the ability to contribute to this book. I hope that it enables others to learn the details about the kernel, discover that driver development is not a scary or forbidding place, and possibly encourage others to join in and help in the collective effort of making this operating system work on every computing platform with every type of device available. The development procedure is fun, the community is rewarding, and everyone benefits from the effort involved.

Now it's back to making this edition obsolete by fixing current bugs, changing APIs to work better and be simpler to understand for everyone, and adding new features. Come along; we can always use the help.

Audience for This Book

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 is useful as a starting point for people who want to become kernel hackers but don't know where to start.

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.

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. Therefore, it 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.

Organization of the Material

The book introduces its topics in ascending order of complexity and is divided into two parts. The first part (Chapters 1–11) 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 quick summary 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 Chapters 1 and 1, however, we 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 (Chapters 12–18) describes block drivers and network interfaces and goes deeper into more advanced topics, such as working with the virtual memory subsystem and with the PCI and USB buses. Many driver authors do not need all of 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. Some Unix expertise is needed as well, as we often refer to Unix semantics about system calls, 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 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.

Online Version and License

The authors have chosen to make this book freely available under the Creative Commons "Attribution-ShareAlike" license, Version 2.0:

http://www.oreilly.com/catalog/linuxdrive3

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: "*Linux Device Drivers*, Third Edition, by Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman. Copyright 2005 O'Reilly Media, Inc., 0-596-00590-3."

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/linuxdrive3

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

Thanks to 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 have laid the philosophical and practical foundations of the current free software renaissance.

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. Reviewers for the third edition were Allan B. Cruse, Christian Morgner, James Bottomley, Jerry Cooperstein, Patrick Mochel, Paul Kinzelman, and Robert Love. 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.

Jon

I must begin by thanking my wife Laura and my children Michele and Giulia for filling my life with joy and patiently putting up with my distraction while working on this edition. The subscribers of LWN.net have, through their generosity, enabled much of this work to happen. The Linux kernel developers have done me a great service by letting me be a part of their community, answering my questions, and setting me straight when I got confused. Thanks are due to readers of the second edition of this book whose comments, offered at Linux gatherings over much of the world,

have been gratifying and inspiring. And I would especially like to thank Alessandro Rubini for starting this whole exercise with the first edition (and staying with it through the current edition); and Greg Kroah-Hartman, who has brought his considerable skills to bear on several chapters, with great results.

Alessandro

I 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. I want to thank Giorgio and Giulia, who have been involved in later editions of the book and happily accepted to be sons of "a gnu" who often works late in the night. I owe a lot to all the free-software authors who actually taught me how to program by making their work available for anyone to study. But for this edition, I'm mostly grateful to Jon and Greg, who have been great mates in this work; it couldn't have existed without each and both of them, as the code base is bigger and tougher, while my time is a scarcer resource, always contended for by clients, free software issues, and expired deadlines. Jon has been a great leader for this edition; both have been very productive and technically invaluable in supplementing my small-scale and embedded view toward programming with their expertise about SMP and number crunchers.

Greg

I would like to thank my wife Shannon and my children Madeline and Griffin for their understanding and patience while I took the time to work on this book. If it were not for their support of my original Linux development efforts, I would not be able to do this book at all. Thanks also to Alessandro and Jon for offering to let me work on this book; I am honored that they let me participate in it. Much gratitude is given to all of the Linux kernel programmers, who were unselfish enough to write code in the public view, so that I and others could learn so much from just reading it. Also, for everyone who has ever sent me bug reports, critiqued my code, and flamed me for doing stupid things, you have all taught me so much about how to be a better programmer and, throughout it all, made me feel very welcome to be part of this community. Thank you.