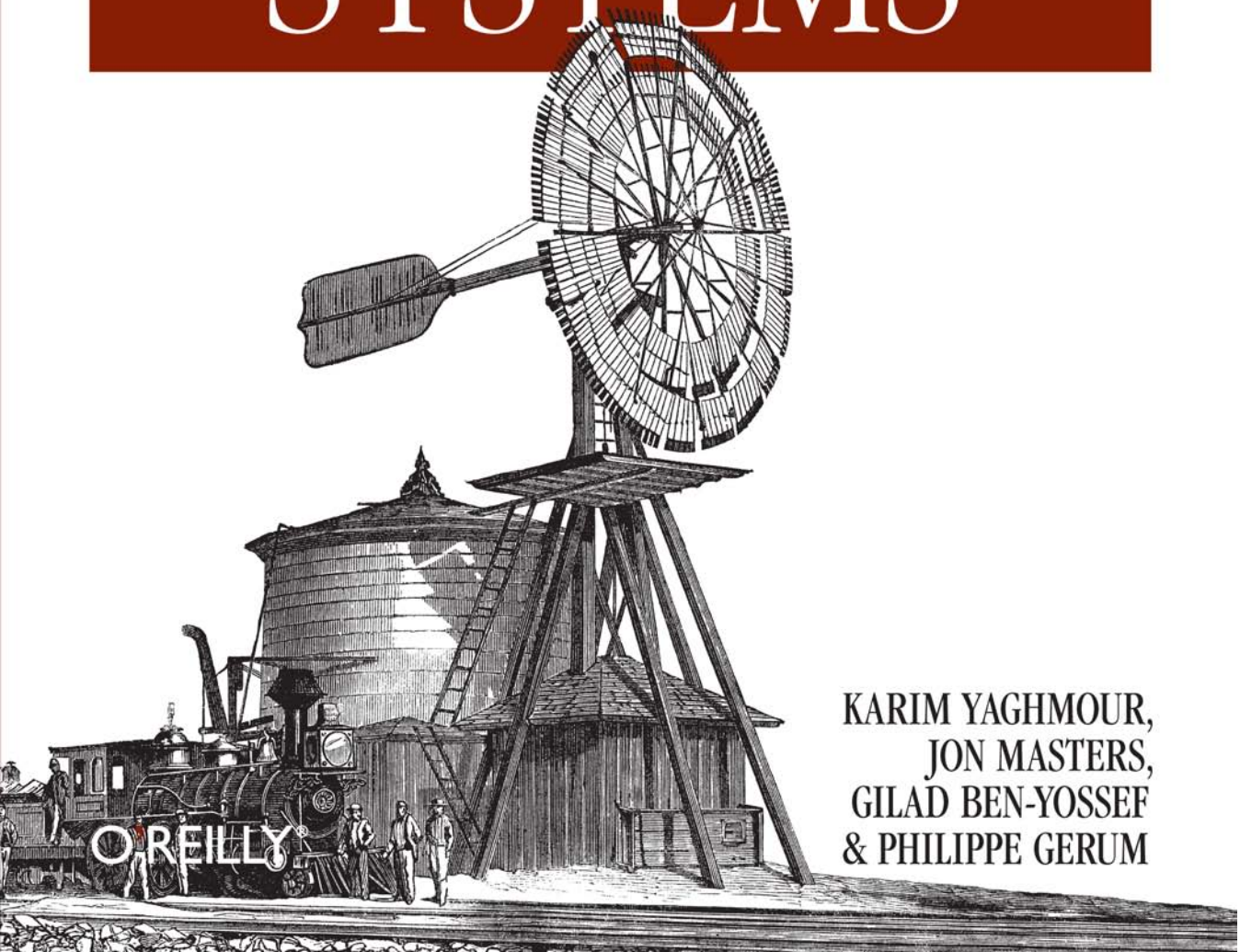
CONCEPTS, TECHNIQUES, TRICKS & TRAPS

2nd Edition
Includes Real-Time Variants

Building Embedded LINUX SYSTEMS



KARIM YAGHMOUR,
JON MASTERS,
GILAD BEN-YOSSEF
& PHILIPPE GERUM

O'REILLY®

Building Embedded Linux Systems

Other Linux resources from O'Reilly

Related titles

- Designing Embedded Hardware
- Linux Device Drivers
- Linux in a Nutshell
- Linux Network Administrator's Guide
- Programming Embedded Systems
- Running Linux
- Understanding the Linux Kernel

Linux Books Resource Center

linux.oreilly.com is a complete catalog of O'Reilly's books on Linux and Unix and related technologies, including sample chapters and code examples.

ONLamp.com is the premier site for the open source web platform: Linux, Apache, MySQL, and either Perl, Python, or PHP.

Conferences

O'Reilly brings diverse innovators together to nurture the ideas that spark revolutionary industries. We specialize in documenting the latest tools and systems, translating the innovator's knowledge into useful skills for those in the trenches. Visit *conferences.oreilly.com* for our upcoming events.

Safari Bookshelf (*safari.oreilly.com*) is the premier online reference library for programmers and IT professionals. Conduct searches across more than 1,000 books. Subscribers can zero in on answers to time-critical questions in a matter of seconds. Read the books on your Bookshelf from cover to cover or simply flip to the page you need. Try it today for free.

SECOND EDITION

Building Embedded Linux Systems

Karim Yaghmour, Jon Masters, Gilad Ben-Yossef, and Philippe Gerum

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

Building Embedded Linux Systems, Second Edition
by Karim Yaghmour, Jon Masters, Gilad Ben-Yossef, and Philippe Gerum

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Editor: Andy Oram
Production Editor: Loranah Dimant
Copyeditor: Genevieve d'Entremont
Proofreader: Loranah Dimant

Indexer: Joe Wizda
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Jessamyn Read

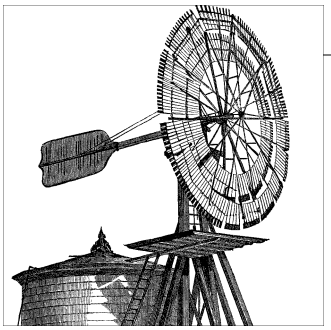
Printing History:
April 2003: First Edition.
August 2008: Second Edition.

ISBN: 978-0-596-52968-0

[M]

1218037492

Table of Contents



Preface

When the author of this book's first edition, Karim Yaghmour, first suggested using Linux in an embedded system back in 1997 while working for a hardware manufacturer, his suggestion was met with a certain degree of skepticism and surprise. Today, Linux is either in use already or is being actively considered for most embedded systems. Indeed, many industry giants and government agencies are increasingly relying on Linux for their embedded software needs.

This book was very well received in its first edition, but a number of advances in the Linux kernel and accompanying tools since the book's appearance make Linux even more attractive. Foremost among these are a number of real-time extensions and companion environments, some of which are discussed in the last three chapters of this edition.

Also, since the first edition of this book, enthusiastic open source and free software programmers have simplified the building and installation of GNU/Linux components (we use "GNU" here to acknowledge the centrality of tools from this free software project in creating functional Linux systems). This second edition therefore introduces you to a world of wonderful high-level tools, including Eclipse and various tools that "build the build tools" for embedded Linux systems. But we preserve much of the low-level information for those who need it, and to help you understand what the helper tools are doing behind the scenes.

In keeping with the explosions of progress on various parts of Linux and accompanying tools, it's useful to get a variety of expert perspectives on topics in embedded and real-time Linux. Therefore, for the second edition of this book the authors are joined by a number of key participants in the GNU/Linux community, including those doing kernel development or creating related projects.

Focus on Self-Sufficiency

The widespread interest and enthusiasm generated by Linux's successful use in a number of embedded applications has led to the creation of a plethora of articles, websites, companies, and documents all pertaining to "embedded Linux." Yet, beyond the flashy announcements, the magazine articles, and the hundreds of projects and products that claim to ease Linux's use in embedded systems, professional developers seeking a useful guide are still looking for answers to fundamental questions regarding the basic methods and techniques required to build embedded systems based on the Linux kernel.

Much of the documentation currently available relies heavily on the use of a number of prepackaged, ready-to-use cross-platform development tools and target binaries. Yet other documents cover only one very precise aspect of running Linux on an embedded target.

The first edition of this book was a radical departure from the existing documentation in that, other than your desire to use Linux, it makes no assumptions as to the tools you have at hand or the scope of your project. All that is required for this book is an Internet connection to download the necessary packages, browse specific online documentation, and benefit from other developers' experiences, as well as share your own, through project mailing lists. You still need a development host and documentation regarding your target's hardware, but the explanations we outline do not require the purchasing of any product or service from any vendor.

Besides giving the greatest degree of freedom and control over your design, this approach is closest to that followed by the pioneers who have spearheaded the way for Linux's use in embedded systems. In essence, these pioneers have *pulled* on Linux to fit their applications by stripping it down and customizing it to their purposes. Linux's penetration of the embedded world contrasts, therefore, with the approach followed by many software vendors to *push* their products into new fields of applications. As an embedded system developer, you are likely to find Linux much easier to pull toward your design than to adapt the products being pushed by vendors to that same design.

This book's approach is to allow you to pull Linux toward your design by providing all the details and discussing many of the corner cases encountered when using Linux in embedded systems. Although it is not possible to claim that this book covers all embedded designs, the resources provided here allow you to easily obtain the rest of the information required for you to customize and use Linux in your embedded system.

In writing this book, our intent was to bring the embedded system developers who use open source and free software in their designs closer to the developers who create and maintain these open source and free software packages. Though a lot of mainstream embedded system developers—many of whom are high-caliber programmers—rely on third-party offerings for their embedded Linux needs, there is a clear opportunity for them to contribute to the open source and free software projects on which they rely.

Ultimately, this sort of dynamic will ensure that Linux continues to be the best operating system choice for embedded systems.

Audience for This Book

This book is intended first and foremost for the experienced embedded system designer who wishes to use Linux in a current or future project. Such a reader is expected to be familiar with all the techniques and technologies used in developing embedded systems, such as cross-compiling, BDM or JTAG debugging, and the implications of dealing with immature or incomplete hardware. If you are such a reader, you may want to skip some of the background material about embedded system development presented early in some sections. There are, however, many early sections (particularly in Chapter 2) that you will need to read, because they cover the special implications of using the Linux kernel in an embedded system.

This book is also intended for the beginning embedded system developer who would like to become familiar with the tools and techniques used in developing embedded systems based on Linux. This book is not an introduction to embedded systems, however, and you may need to research some of the issues discussed here in an introductory textbook.

If you are a power user or a system administrator already familiar with Linux, this book should help you produce highly customized Linux installations. If you find that distributions install too many packages for your liking, for example, and would like to build your own custom distribution from scratch, many parts of this book should come in handy, particularly Chapter 6.

Finally, this book should be helpful to a programmer or a Linux enthusiast who wants to understand how Linux systems are built and operated. Though the material in this book does not cover how general-purpose distributions are created, many of the techniques covered here apply, to a certain extent, as much to general purpose distributions as they do to creating customized embedded Linux installations.

Scope and Background Information

To make the best of Linux's capabilities in embedded systems, you need background in all of the following topics (which are treated distinctly in many books):

Embedded systems
: You need to be familiar with the development, programming, and debugging of embedded systems in general, from both the software and hardware perspectives.

Unix system administration
: You need to be able to tend to various system administration tasks such as hardware configuration, system setup, maintenance, and using shell scripts to automate tasks.

Linux device drivers
: You need to know how to develop and debug various kinds of Linux device drivers.

Linux kernel internals
: You need to understand as much as possible how the kernel operates.

GNU software development tools
: You need to be able to make efficient use of the GNU tools. This includes understanding many of the options and utilities often considered to be "arcane."

We assume that you are familiar with at least the basic concepts of each topic. On the other hand, you don't need to know how to create Linux device drivers to read this book, for example, or know everything about embedded system development. As you read through this book and progress in your use of Linux in embedded systems, you will likely feel the need to obtain more information regarding certain aspects of Linux's use.

Though this book discusses only the use of Linux in embedded systems, part of this discussion can certainly be useful to developers who intend to use one of the BSD variants in their embedded system. Many of the explanations included here will, however, need to be reinterpreted in light of the differences between BSD and Linux.

Organization of the Material

There are four major parts to this book. The first part is composed of Chapters 1 through 3. These chapters cover the preliminary background required for building any sort of embedded Linux system. Though they describe no hands-on procedures, they are essential to understand many aspects of building embedded Linux systems.

The second part spans Chapters 4 through 9. These important chapters lay out the essential steps involved in building any embedded Linux system. Regardless of your system's purpose or functionality, these chapters are required reading.

The third part of the book, which ended the first edition, is made up of Chapters 10 and 11 and covers material that, athough very important, is not essential to building embedded Linux systems.

The final part of the book, comprised of Chapters 12 through 14, is an in-depth discussion of real-time, including its different applications and when you should consider the various implementations and varieties available. We are lucky and honored to have chapters written by the implementors of the Xenomai cokernel and the RT patch to the Linux kernel.

Chapter 1, *Introduction*, gives an in-depth introduction to the world of embedded Linux. It lays out basic definitions and then introduces real-life issues about embedded Linux systems, including a discussion of open source and free software licenses from the embedded perspective. The chapter then introduces the example system used in other parts of this book and the implementation method used throughout the book.

Chapter 2, *Basic Concepts*, outlines the basic concepts that are common to building all embedded Linux systems.

Chapter 3, *Hardware Support*, provides a thorough review of the embedded hardware supported by Linux, and gives links to websites where the drivers and subsystems implementing this support can be found. This chapter discusses processor architectures, buses and interfaces, I/O, storage, general-purpose networking, industrial grade networking, and system monitoring.

Chapter 4, *Development Tools*, covers the installation and use of the various development tools used in building embedded Linux systems. This includes a discussion of Eclipse for embedded Linux development, and how to build and install the GNU toolchain components from scratch. It also includes sections discussing Java, Perl, Python, and other languages, along with a section about the various terminal emulators that can be used to interact with an embedded target.

Chapter 5, *Kernel Considerations*, discusses the selection, configuration, cross-compiling, installation, and use of the Linux kernel in an embedded system.

Chapter 6, *Root Filesystem Content*, updated for the second edition by Michael Opdenacker, explains how to build a root filesystem using the components introduced earlier in the book, including the installation of the C library and the creation of the appropriate */dev* entries. More importantly, this chapter covers the installation and use of BusyBox, embutils, and System V *init*.

Chapter 7, *Storage Device Manipulation*, updated for the second edition by kernel developer David Woodhouse, covers the intricacies of manipulating and setting up storage devices for embedded Linux systems. The chapter's emphasis is on solid-state storage devices, such as native flash and DiskOnChip devices, and the MTD subsystem.

Chapter 8, *Root Filesystem Setup*, explains how to set up the root filesystem created in Chapter 6 for the embedded system's storage device. This includes the creation of filesystem images (based on JFFS2, CRAMFS, or other specialized filesystems), and the use of disk-style filesystems over NFTL.

Chapter 9, *Setting Up the Bootloader*, discusses the various bootloaders available for use in each embedded Linux architecture. Special emphasis is put on the use of GRUB with DiskOnChip devices and U-Boot. Network booting using BOOTP/DHCP, TFTP, and NFS is also covered.

Chapter 10, *Setting Up Networking Services*, focuses on the configuration, installation, and use of software packages that offer networking services, such as SNMP, SSH, and HTTP.

Chapter 11, *Debugging Tools*, updated for the second edition by Michael Boerner, covers the main debugging issues encountered in developing software for embedded Linux systems. This includes the use of *gdb* in a cross-platform development environment, Eclipse, tracing, performance analysis, and memory debugging.

Chapter 12, *Introduction to Real-Time Linux*, explains the value of real-time and offers a candid discussion of when you need various real-time features, along with an introduction to the various ways you can achieve real-time behaviors using Linux. This chapter was written by the founder and maintainer of the Xenomai Real-Time System, Philippe Gerum.

Chapter 13, *The Xenomai Real-Time System*, also written by Philippe Gerum, offers a high-level view of how Xenomai achieves real-time goals and how it can be useful in conjunction with embedded Linux.

Chapter 14, *The RT Patch*, performs a similar function for the RT patch to the Linux kernel, explaining how to enable its features. The chapter was written by Steven Rostedt, a key developer on the patch.

Although Chapters 7 through 9 are independent, note that their content is highly interrelated. For example, setting up the target's storage device, as discussed in Chapter 7, requires a basic knowledge about the target filesystem organization as discussed in Chapter 8, and vice versa. So, too, does setting up storage devices require a basic knowledge of bootloader setup and operation as discussed in Chapter 9, and vice versa. We therefore recommend that you read Chapters 7 through 9 in one breath a first time before carrying out the instructions in any of them. When setting up your target thereafter, you will nevertheless follow the same sequence of operations outlined in these chapters.

Hardware Used in This Book

As you'll see in Chapter 3, Linux supports a very wide range of hardware. For this book, we've used a number of embedded systems to test the various procedures. Some of these systems, such as the OpenMoko-based NEO 1973, are commercial products available in the mainstream market. We included these intentionally, to demonstrate that any willing reader can find the materials to support learning how to build embedded Linux systems. You can, of course, still use an old x86 PC for experimenting, but you are likely to miss much of the fun, given the resemblance between such systems and most development hosts.

To illustrate the range of target architectures on which Linux can be used, we varied the target hardware we used in the examples between chapters. Though some chapters are based on different architectures, the commands given in each chapter apply readily to other architectures as well. If, for instance, an example in a chapter relies on the *arm-linux-gcc* command, which is the *gcc* compiler for ARM, the same example would work for a PPC target by using the *powerpc-linux-gcc* command instead. Whenever more than one architecture is listed for a chapter, the main architecture discussed is the first one listed. The example commands in Chapter 5, for instance, are mainly centered around PowerPC, but there are also a few references to ARM commands.

Unless specific instructions are given to the contrary, the host's architecture is always different from the target's. In Chapter 4, for example, we used a PPC host to build tools for an x86 target. The same instructions could, nevertheless, be carried out on a SPARC or an S/390 with little or no modification. Note that most of the content of the early chapters is architecture- independent, so there is no need to provide any architecture-specific commands.

Software Versions

The central software on which an embedded Linux system depends, of course, is the Linux kernel. This book concentrates on version 2.6 of the Linux kernel, and 2.6.22 in particular. Changes to the kernel will probably have only a benign effect on the information in the book. That is, new releases will probably support more hardware than Chapter 3 lists. But the essential tasks described in this book are unlikely to change.

In addition, this book discusses the configuration, installation, and use of over 40 different open source and free software packages. Each package is maintained independently and is developed at a different pace. Because these packages change over time, it is likely that the package versions covered in this book may be outdated by the time you read it. In an effort to minimize the effect of software updates on the text, we have kept the text as version-independent as possible. The overall structure of the book and the internal structure of each chapter, for example, are unlikely to vary regardless of the various software changes. Also, many packages covered in this book have been around for quite some time, so they are unlikely to change in any substantial way. For instance, the commands to install, set up, and use the different components of the GNU development toolchain, which is used throughout this book, have been relatively constant for a number of years and are unlikely to change in any substantial way in the future. This statement applies equally to most other software packages discussed.

Typographical Conventions

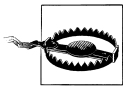
The following is a list of typographical conventions used in this book:

`Constant width`
: Used to show the contents of code files or the output from commands, and to indicate source code keywords that appear in code.

`Constant width bold`
: Used to indicate user input.

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

This icon indicates a tip, suggestion, or general note.



This icon indicates a warning or caution.

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. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Building Embedded Linux Systems*, by Karim Yaghmour, Jon Masters, Gilad Ben-Yossef, and Philippe Gerum. Copyright 2008 Karim Yaghmour and Jon Masters, 978-0-596-52968-0."

Contact Information

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

O'Reilly Media.
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, or any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596529680

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 website at:

http://www.oreilly.com

The authors also have a site for this book at:

http://www.embeddedlinuxbook.org/

Safari® Books Online

When you see a Safari® Books Online 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 for the First Edition

E quindi uscimmo a riveder le stelle.[*] It is with these words that Dante ends *Inferno*, the first part of his *Divine Comedy*. Though it would be misleading to suggest that writing this book wasn't enjoyable, Dante's narrative clearly expresses the feeling of finishing a first iteration of the book you now hold in your hands. In particular, I have to admit that it has been a challenging task to pick up the bits and pieces of information available on the use of Linux in embedded systems, to complete this information in as much as possible, and put everything back together in a single, straightforward manuscript that provides a practical method for building embedded Linux systems. Fortunately, I was aided in this task by very competent and willing people.

First and foremost, I would like to thank Andy Oram, my editor. Much like Virgil assisted Dante in his venture, Andy shepherded me throughout the various stages of writing this book. Among many other things, he patiently corrected my nonidiomatic phrases, made sure that my text actually conveyed the meaning I meant for it to convey, and relentlessly pointed out the sections where I wasn't providing enough detail. The text you are about to read is all the much better, as it has profited from Andy's input. By the same token, I would like to thank Ellen Siever, with whom I initially started working on this book. Though our collaboration ended earlier than I wished it had, many of the ideas that have made their way into this final version of the book have profited from her constructive feedback.

I have been extremely fortunate to have an outstanding team of reviewers go over this book, and am very grateful for the many hours they poured into reading, correcting, and pointing out problems with various aspects of this book. The review team was

[*] "And from there we emerged to see the stars once more."

made up of Erik Andersen, Wolfgang Denk, Bill Gatliff, Russell King, Paul Kinzelman, Alessandro Rubini, David Schleef, and David Woodhouse. I'd like to especially thank Alessandro for his dogged pursuit of perfection. Any remaining errors you may find in the following pages are without a doubt all mine.

Writing about the use of Linux in embedded systems requires having access to a slew of different hardware. Given that embedded hardware is often expensive, I would like to thank all the companies and individuals who have stepped forward to provide me with the appropriate equipment. In particular, I would like to thank Stéphane Martin of Kontron for providing a Teknor VIPer 806 board, Wolfgang Denk of DENX Software Engineering for providing a TQ components TQM860L PPC board, and Steve Papacharalambous and Stuart Hughes of Zee2 for providing a uCdimm system.

I have found much of the incentive and thrust for writing this book from being a very satisfied open source and free software user and contributor who has profited time and again from the knowledge and the work produced by other members of this community. For this, I have many people to thank. Primarily, I'd like to thank Michel Dagenais for his trust, his guidance, and for giving me the chance to freely explore uncharted terrain. My work on developing the Linux Trace Toolkit, as part of my masters degree with Michel, got me more and more involved in the open source and free software community. As part of this involvement, I have met a lot of remarkable individuals whose insight and help I greatly appreciate. Lots of thanks to Jacques Gélinas, Richard Stallman, Jim Norton, Steve Papacharalambous, Stuart Hughes, Paolo Mantegazza, Pierre Cloutier, David Schleef, Wolfgang Denk, Philippe Gerum, Loic Dachary, Daniel Phillips, and Alessandro Rubini.

Last, but certainly not least, I owe a debt of gratitude to Sonia for her exceptional patience as I spent countless hours testing, writing, testing some more, and writing even more. Her support and care have made this endeavor all the more easy to carry out. *La main invisible qui a écrit les espaces entre les lignes est la sienne et je lui en suis profondément reconnaissant*.†

Acknowledgments for the Second Edition

When Karim first mentioned updating *Building Embedded Linux Systems*, I could not have imagined what a fun and wild ride it would be. I was in the final stages of moving from the U.K. to the U.S. at the time, and life was pretty hectic for quite a while. Along the way, some great friends and coauthors have helped to turn an idea into the reality of the book that you are now reading. And we collectively hope that we have served to increase the range of documentation available on embedded Linux.

† "The invisible hand that wrote the spaces between each line is hers, and I am profoundly grateful to her for this."

First and foremost, I would like to thank my friend Karim Yaghmour for letting me run amock with his original manuscript, Andy Oram for his patient advice and editorial wizardry, and Isabel Kunkle for assisting Andy in putting up with a bunch of authors with busy schedules. I would also like to thank Marlowe Shaeffer and the team at O'Reilly for their steadfast attention to detail, especially near the end of the project.

I would like to thank my coauthors for stepping up to the plate and helping to see this project through: Michael Boerner, Michael Opdenacker, Steven Rostedt, Gilad Ben-Yossef (CTO, Codefidence Ltd.), Phillipe Gerum, and David Woodhouse. I've known most of you for many years, even if we only get to meet once a year at the Linux Symposium, and I am grateful that you have helped to improve the overall quality of this book. In a similar vain, I am grateful to the review comments from Tim Rikers, Vince Skahan, and Mark VandenBrink, as well as the many others I have occasionally spoken with about this book. But all that said, any remaining mistakes and technical omissions are entirely my responsibility, though we hope there are few.

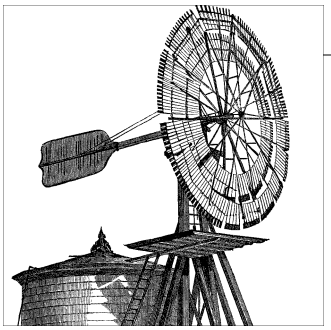
Embedded Linux would mean nothing without the hard work of many thousands of people all over the world. Some of those people have gotten involved in the first or second editions of this book, while there are many, many more people out there helping to make Linux the most valuable and viable choice for embedded developers. It would be tricky to even attempt to list these people by name, and so I would like to instead offer my most sincere thanks to everyone concerned—I'd also like to encourage readers to thank those who provide the upstream for their development projects. Please do also encourage your employers and customers to do the same through whatever means you feel is most appropriate.

I would like to thank my friends and family for their never-ending support of my many pursuits and random craziness. My mum and dad rarely see me these days (I live 3,000 miles away in another country, on an awkward time delay) but have always been the best parents you could wish for, in spite of their son turning out to be a "traitor's dog" (thanks, dad, for your whimsical historical insight right there!) who joined the Americans. My sister Hannah and brother-in-law Joe Wrigley (another Red Hatter!) have always been amazing, as has my youngest sister Holly. My grandmother keeps me informed of family goings on with her letters, which I always look forward to reading far away from a computer.

Many friends contributed to the overall success of this project without even realizing it. They include Deepak Saxena, Hussein Jodiyawalla, Bill Weinberg, Alison Cornish, Grace Mackell, Andrew Schliep, Ginger Diercks, Kristin Mattera and James Saunders, Karen Hopkins, Andrew Hutton, and Emilie Moreau (and also Denali and Nihao), Madeleine and Chris Ball, Tim Burke, Lon Hohberger, Chris Lumens, Jon Crowe, Rachel Cox, Catherine Nolan, Toby Jaffey (and Sara and Milly), David Brailsford, Jeff and Nicole Stern, Catherine Davis, Mary-Kay and Luke Jensen, Philippe De Swert, Matt Domsch, Grant Likely (of Secret Lab), Hetal Patel, Mark Lord, Chris Saul, Dan Scrase, and David Zeuthen. A special thanks to Sven-Thorsten Dietrich and Aaron Nielson for their like-minded craziness at just the right moments.

Finally, I am very grateful to my good friend David Brailsford of the University of Nottingham, and to Malcolm Buckingham and Jamie McKendry of Oxford Instruments for believing in me and letting me experiment with Linux and superconducting magnets, and to Ian Graham of MontaVista UK Ltd. for the opportunity to work on some great projects during my time there. I also owe Andrew Hutton and Craig Ross of Steamballoon (and organizers of Linux Symposium) thanks for their support of my embedded endeavors over the years. I would especially like to thank Gary Lamb (Global Engineering Services—our embedded team), Clark Williams, and Tim Burke of Red Hat, Inc. for their continued support, as well as all of my friends at Red Hat and at other great Linux companies.

—Jon Masters, Cambridge, Massachusetts



CHAPTER 1
Introduction

Linux was first released into an unsuspecting world in the summer of 1991. Initially the spare-time hobby of a Finnish computer scientist by the name of Linus Torvalds, Linux was at first accessible only in software source code form to those with enough expertise to build and install it. Early enthusiasts (most also developers themselves by necessity) exploited the growth of the Internet in the early 1990s as a means to build online communities and drive development forward. These communities helped to build the first Linux software distributions, containing all the software components needed to install and use a Linux system without requiring users to be technical experts.

Over the next decade, Linux grew into the mature Unix-like operating system it is today. Linux now powers anything and everything from the smallest handheld gadget to the largest supercomputing cluster, and a nearly infinite range of different devices in between. Examples of the wide range of Linux use abound all around: digital TV receivers and recorders such as TiVo, cell phones from big names like Motorola, Hollywood's huge Linux "render farms" (used to generate many of the recent CGI movies we have seen), and household name websites such as Google. In addition, a growing number of multinational corporations have successfully built businesses selling Linux software.

In many ways, Linux came along at the right moment in time. But it owes a lot of its success to the work of projects that came before it. Without the hard work of Richard Stallman and the Free Software Foundation (FSF) over the decade prior to Linux arriving on the scene, many of the tools needed to actually build and use a Linux system would not exist. The FSF produced the GNU C Compiler (GCC) and many of the other tools and utilities necessary for building your own embedded Linux systems from scratch, or at least from pre-built collections of these tools that are supplied by third-party vendors. Software maintained by the Free Software Foundation comprises a collection known as GNU, for "GNU's Not UNIX," also known (to some) as the GNU system. This stemmed from the FSF's stated goal to produce a free Unix-like system.

Embedded systems running Linux are the focus of this book. In many ways, these are even more ubiquitous than their workstation and server counterparts—mostly due to the sheer volume of devices and consumer gadgets that rely upon Linux for their operation. The embedded space is constantly growing with time. It includes obvious examples, such as cellular telephones, MP3 players, and a host of digital home entertainment devices, but also less-obvious examples, such as bank ATMs, printers, cars, traffic signals, medical equipment, technical diagnostic equipment, and many, many more. Essentially, anything with a microprocessor that is not considered a "computer" but performs some kind of function using computing is a form of embedded system.

If you are reading this book, you probably have a basic idea why one would want to run an embedded system using Linux. Whether because of its flexibility, its robustness, its price tag, the community developing it, or the large number of vendors supporting it, there are many reasons for choosing to build an embedded system with Linux and many ways to carry out the task. This chapter provides the background for the material presented in the rest of the book by discussing definitions, real-life issues, generic embedded Linux systems architecture, and methodology. This chapter sets the stage for later chapters, which will build upon concepts introduced here.

Definitions

The words "Linux," "embedded Linux," and "real-time Linux" are often used with little reference to what is actually being designated with such terminology. Sometimes, the designations may mean something very precise, whereas other times, a broad range or a category of application is meant. In this section, you will learn what the use of these terms can mean in a variety of different situations—starting with the many meanings of "Linux."

What Is Linux?

Technically speaking, Linux refers only to an operating system kernel originally written by Linus Torvalds. The Linux kernel provides a variety of core system facilities required for any system based upon Linux to operate correctly. Application software relies upon specific features of the Linux kernel, such as its handling of hardware devices and its provision of a variety of fundamental abstractions, such as virtual memory, tasks (known to users as processes), sockets, files, and the like. The Linux kernel is typically started by a bootloader or system firmware, but once it is running, it is never shut down (although the device itself might temporarily enter a low-powered suspended state). You will learn more about the Linux kernel in Chapter 5.

These days, the term "Linux" has become somewhat overloaded in everyday communication. In large part, this is due to its growing popularity—people might not know what an operating system kernel is or does, but they will have perhaps heard of the term Linux. In fact, Linux is often used interchangeably in reference to the Linux kernel