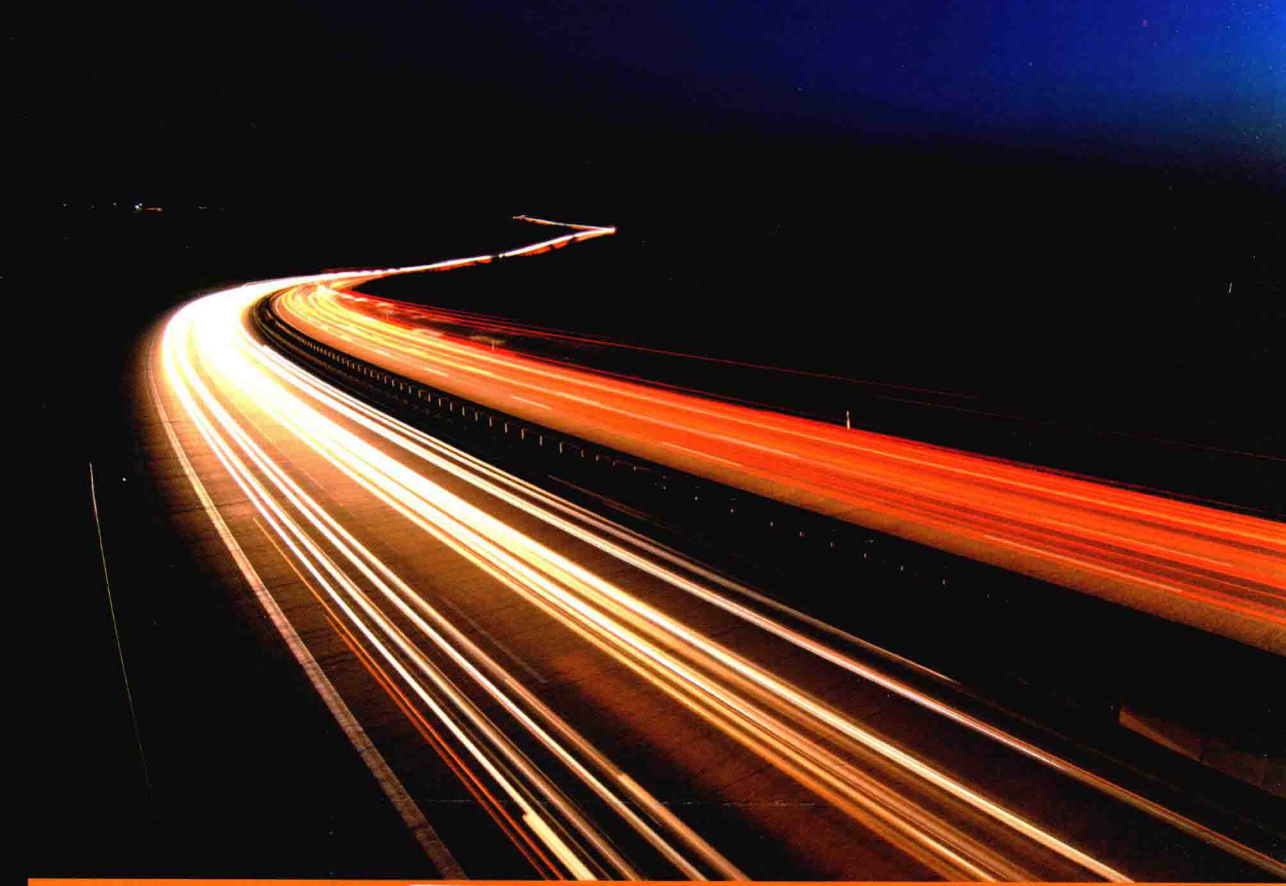

精通嵌入式Linux编程（影印版）

Mastering Embedded Linux Programming

Chris Simmonds 著

[PACKT] PUBLISHING

东南大学出版社 SOUTHEAST UNIVERSITY PRESS

精通嵌入式 Linux 编程(影印版)

Chris Simmonds 著

南京　东南大学出版社

图书在版编目(CIP)数据

精通嵌入式 Linux 编程:英文/(英)克里斯·西蒙兹(Chris Simmonds)著. —影印本. —南京:东南大学出版社,2017.4

书名原文:Mastering Embedded Linux Programming

ISBN 978-7-5641-7078-3

Ⅰ.①精… Ⅱ.①克… Ⅲ.①Linux 操作系统—程序设计—英文 Ⅳ.①TP316.85

中国版本图书馆 CIP 数据核字(2017)第 051863 号

精通嵌入式 Linux 编程(影印版)

出版发行:东南大学出版社
地　　址:南京四牌楼 2 号　　邮编:210096
出 版 人:江建中
网　　址:http://www.seupress.com
电子邮件:press@seupress.com
印　　刷:常州市武进第三印刷有限公司
开　　本:787 毫米×980 毫米　　16 开本
印　　张:26
字　　数:509 千字
版　　次:2017 年 4 月第 1 版
印　　次:2017 年 4 月第 1 次印刷
书　　号:ISBN 978-7-5641-7078-3
定　　价:76.00 元

本社图书若有印装质量问题,请直接与营销部联系。电话(传真):025-83791830

Credits

Author
Chris Simmonds

Reviewers
Robert Berger
Tim Bird
Mathieu Deschamps
Mark Furman
Klaas van Gend
Behan Webster

Commissioning Editor
Kartikey Pandey

Acquisition Editor
Sonali Vernekar

Content Development Editor
Samantha Gonsalves

Technical Editor
Dhiraj Chandanshive

Copy Editor
Kevin McGowan

Project Coordinator
Sanchita Mandal

Proofreader
Safis Editing

Indexer
Priya Sane

Production Coordinator
Manu Joseph

Cover Work
Manu Joseph

Foreword

Linux is an extremely flexible and powerful operating system and I suspect we've yet to truly see it used to full advantage in the embedded world. One possible reason is that there are many different facets to it and the learning curves can be steep and time consuming.

Its possible to figure your way own through the world of Embedded Linux, as I myself have done over the past decade, however I am pleased to see people like Chris putting together books like this which give people a good grounding on many useful topics. I certainly could have used a guide like this back when I started!

I obviously have a personal bias to the Yocto Project, it being my major contribution and attempt to make a difference to the Embedded Linux world. One of its core objectives is to try and make things easier for people building Embedded Linux systems. We've had some successes; there are also areas we know work is still needed. We're continually trying to simplify barriers to entry and let more people get involved, make the technology more accessible and encourage adoption.

In writing this book, Chris is supporting the same objectives. I hope you enjoy the book, enjoy Linux and that ultimately that we might see you becoming a part of the vibrant open source communities that make up many of the components you're about to learn about.

Richard Purdie
Yocto Project Architect, Linux Foundation Fellow

About the Author

Chris Simmonds is a software consultant and trainer who lives in southern England. He has been using Linux in embedded systems since the late 1990s, during which he has worked on many interesting projects, including a stereoscopic camera, intelligent weighing scales, various set-top boxes and home routers, and even a large walking robot.

He is a frequent presenter at open source and embedded conferences, including the Embedded Linux Conference, Embedded World, and the Android Builders' Summit. He has been conducting training courses and workshops in embedded Linux since 2002 and in embedded Android since 2010. He has delivered hundreds of sessions to many well-known companies. You can see some of his work on the "Inner Penguin" blog at `www.2net.co.uk`.

I would like to thank my editor, Samantha Gonsalves, for her tireless work in reviewing my work and keeping me on track. I would also like to thank the people who took time to review my early drafts and to see through my obfuscations to the core of what I was trying to say. So, I would like to thank Behan Webster, Klaas van Gend, Tim Bird, Robert Berger, Mathieu Deschamps, and Mark Furman. Last but not least, I would like to thank my wife, Shirley Simmonds, for her support and for understanding the fact that I really could not help her redecorate the house because I had a book to write.

About the Reviewers

Robert Berger has been gathering practical and managerial experience in software design and development for embedded systems with and without hard real-time requirements since 1993. Since the beginning of the 21st century, he has been using GNU/Linux on desktops and server class machines, but mainly for embedded practices (automotive, industrial control, robotics, telecoms, consumer electronics, and so on). He regularly attends international events, such as Embedded World, Embedded Software Engineering Congress, Embedded Systems Conference, and Embedded Linux Conference as an expert and lecturer. His specializes mainly in training, but also in consulting (in German or English) worldwide. Robert's expertise ranges from the smallest real-time systems (FreeRTOS) to setups with multiple processors/cores and embedded GNU/Linux (user-, kernel-space, device drivers, hardware interfacing, debugging, multi-core, and the Yocto project) with a focus on free and open source software. He is a globetrotter. He is the CEO and embedded software specialist at Reliable Embedded Systems, which is based in St. Barbara, Austria. When he is not traveling, he lives with his family in Athens, Greece. Feel free to contact him on his website at `http://www.ReliableEmbeddedSystems.com`.

He has reviewed the book *Embedded Linux Systems with the Yocto Project* (*Prentice Hall Open Source Software Development Series*) by *Rudolf J. Streif*.

Tim Bird works as a senior software engineer for Sony Mobile Communications, where he helps Sony improve the Linux kernel for use in Sony's products. He is also the chair of the Architecture Group of the CE Working Group of the Linux Foundation. He has been working with Linux for over 20 years. He helped found two different embedded Linux trade associations and is the creator of the Embedded Linux Conference, which he still leads. Earlier in his embedded Linux career, Tim coauthored the book *Using Caldera OpenLinux*.

Mathieu Deschamps is the founder of ScourGE (`www.scourge.fr`), which provides open source software/hardware innovation services to its clients. They are leaders in the fields of telecommunication, mobile communication, industrial processes, and decision support systems.

He is an R&D business consultant and a trainer. Also, since 2003, he has been an independent tech-driver, involved in many large and small scale projects around GNU/Linux, Android, embedded systems engineering, and security.

Mark Furman, author of OpenVZ Essentials, currently works as a systems engineer for Info-Link Technologies. He has been in the IT field for over 10 years and specializes in Linux and other open source technologies. In his spare time, he enjoys writing and reviewing books on Linux and other open source technologies as well as tinkering with Arduino, Python, and Raspberry Pi projects at Knox Labs, a Hackerspace located in Knox County, OH.

Klaas van Gend graduated in systems and control engineering at Eindhoven University of Technology in the Netherlands. He worked for companies, including Philips, Siemens, and Bosch, writing software for printer prototypes, video encryption, car infotainment, medical equipment, industrial automation, and navigation systems. In 2004, he switched over to MontaVista Software, which was the market leader for embedded Linux. As a systems architect and consultant, he worked with many companies all over Europe, integrating embedded Linux into their products.

For the last few years, he has been working as a trainer and consultant for Vector Fabrics, a small start-up specializing in multi-core programming and software dynamic analysis. He teaches multi-core programming in C and C++ and helps customers improve software performance by utilizing hardware resources in a better way. Vector Fabrics' Pareon tool suite also automatically finds hard-to-find dynamic bugs, including data races, buffer overruns, use-after-free for heap and stack variables, and memory leaks.

He has authored over 100 magazine articles and papers on (embedded) Linux, programming, performance, systems design, and computer games. He cofounded the Embedded Linux Conference Europe and was a lead developer for several open source projects, including UMTSmon for 3G cellular networks and the physics puzzle game *The Butterfly Effect*.

When not at work, he reads urban fantasy or can be found at Aeroclub Nistelrode, piloting glider planes.

Behan Webster has spent two decades in diverse tech industries such as telecom, datacom, optical, embedded, and automotive writing code for a range of hardware from the very small to the very large. His Linux experience spans kernel programming, Embedded Linux, and board bring-up. Currently, he is the lead consultant at Converse in Code Inc, an embedded Linux engineer and project lead working on the LLVMLinux project as well as being a trainer for The Linux Foundation.

www.PacktPub.com

Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit `www.PacktPub.com`.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`https://www2.packtpub.com/books/subscription/packtlib`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

Preface

An embedded system is a device with a computer inside that doesn't look like a computer. Washing machines, televisions, printers, cars, aircraft, and robots are all controlled by a computer of some sort, and in some cases, more than one. As these devices become more complex, and as our expectations of the things that we can do with them expand, the need for a powerful operating system to control them grows. Increasingly, Linux is the operating system of choice.

The power of Linux stems from its open source model, which encourages sharing of code. This means that software engineers from many backgrounds, and often employed by competing companies, can cooperate to create an operating system kernel that is up-to-date and tracks the development of the hardware. From this one code base, there is support from the largest super computers down to a wristwatch. Linux is only one component of the operating system. Many other components are needed to create a working system, from basic tools, such as a command shell, to graphical user interfaces, with web content and communicating with cloud services. The Linux kernel together with an extensive range of other open source components allow you to build a system that can function in a wide range of roles.

However, flexibility is a double-edged sword. While it gives a system designer a wide choice of solutions to a particular problem, it also presents the problem of knowing which are the best choices. The propose of this book is to describe in detail how to construct an embedded Linux system using free, open source projects to produce a robust, reliable, and efficient system. It is based on the experience of the author as a consultant and trainer over a period of many years, using examples to illustrate best practices.

What this book covers

Mastering Embedded Linux Programming is organized along the lines of the life cycle of a typical embedded Linux project. The first six chapters tell you what you need to know about how to set up the project and how a Linux system is put together, culminating in selecting an appropriate Linux build system. Next, comes the stage where certain key decisions must be made about the system architecture and design choices, including flash memory, device drivers, and the `init` system. Following this is the phase of writing applications to make use of the embedded platform you have built, and for which there are two chapters on processes, threads, and memory management. Finally, we come to the stage of debugging and optimizing the platform, which is discussed in chapters 12 and 13. The last chapter describes how to configure Linux for real-time applications.

Chapter 1, Starting Out, sets the scene by describing the choices available to the system designer at the start of a project.

Chapter 2, Learning About Toolchains, describes the components of a toolchain with an emphasis on cross-compiling. It describes where to get a toolchain and provides details on how to build one from the source code.

Chapter 3, All About Bootloaders, explains the role of the bootloader to initialize the hardware of the device and uses U-Boot and Barebox as examples. It also describes the device tree, which is a means of encoding the hardware configuration, used in many embedded systems.

Chapter 4, Porting and Configuring the Kernel, provides information on how to select a Linux kernel for an embedded system and configure it for the hardware within the device. It also covers how to port Linux to the new hardware.

Chapter 5, Building a Root Filesystem, introduces the ideas behind the user space part of an embedded Linux implementation by means of a step-by-step guide on how to configure a root filesystem.

Chapter 6, Selecting a Build System, covers two embedded Linux build systems, which automate the steps described in the previous four chapters and conclude the first section of the book.

Chapter 7, Creating a Storage Strategy, discusses the challenges created by managing flash memory, including raw flash chips and embedded MMC or eMMC packages. It describes the filesystems that are applicable to each type of technology. It also covers techniques on how to update the device firmware in the field.

Chapter 8, Introducing Device Drivers, describes how kernel device drivers interact with the hardware with worked examples of a simple driver. It also describes the various ways of calling device drivers from the user space.

Chapter 9, Starting up - the init Program, shows how the first user space program, `init`, which starts the rest of the system. It describes the three versions of the `init` program, each suitable for a different group of embedded systems, with increasing complexity from BusyBox `init` to systemd.

Chapter 10, Learning About Processes and Threads, describes embedded systems from the point of view of the application programmer. This chapter looks at processes and threads, inter-process communication, and scheduling policies.

Chapter 11, Managing Memory, introduces the ideas behind virtual memory and how the address space is divided into memory mappings. It also covers how to detect memory that is being used and memory leaks.

Chapter 12, Debugging with GDB, shows you how to use the GNU debugger, GDB, to interactively debug both the user space and kernel code. It also describes the kernel debugger, `kdb`.

Chapter 13, Profiling and Tracing, covers the techniques available to measure the system performance, starting from whole system profiles and then zeroing in on particular areas where bottlenecks are causing poor performance. It also describes Valgrind as a tool to check the correctness of an application's use of thread synchronization and memory allocation.

Chapter 14, Real-time Programming, provides a detailed guide to real-time programming on Linux, including the configuration of the kernel and the real-time kernel patch, and also provides a description of tools to measure real-time latencies. It also covers information on how to reduce the number of page faults by locking the memory.

What you need for this book

The software used in this book is entirely open source. The versions used are, in most cases, the latest stable versions available at the time of writing. While I have tried to describe the main features in a manner that are not specific to a particular version, it is inevitable that the examples of commands contain some details that will not work with the later versions. I hope that the descriptions that accompany them are sufficiently informative so that you can apply the same principles to the later versions of the package.

There are two systems involved in creating an embedded system: the host system that is used to cross-compile the software and the target system on which it runs. For the host system, I have used Ubuntu 14.04, but most Linux distributions will work with little modification. In the same way, I had to choose a target system to represent an embedded system. I chose two: the BeagelBone Black and the QEMU CPU emulator, emulating an ARM target. The latter target means that you can try out the examples without having to invest in the hardware for an actual target device. At the same time, it should be possible to apply the examples to a wide range of targets with adaptations for specifics, such as device names and memory layout.

The versions of the main packages for the target are U-Boot 2015.07, Linux 4.1, Yocto Project 1.8 "Fido", and Buildroot 2015.08.

Who this book is for

This book is ideal for Linux developers and system programmers who are already familiar with embedded systems and who want to know how to create best-in-class devices. A basic understanding of C programming and experience with systems programming is needed.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "We could use the stream I/O functions `fopen(3)`, `fread(3)`, and `fclose(3)`."

A block of code is set as follows:

```
static struct mtd_partition omap3beagle_nand_partitions[] = {
  /* All the partition sizes are listed in terms of NAND block
size */
  {
    .name       = "X-Loader",
    .offset     = 0,
    .size       = 4 * NAND_BLOCK_SIZE,
    .mask_flags  = MTD_WRITEABLE,  /* force read-only */
  }
}
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
static struct mtd_partition omap3beagle_nand_partitions[] = {
  /* All the partition sizes are listed in terms of NAND block
size */
  {
    .name         = "X-Loader",
    .offset       = 0,
    .size          = 4 * NAND_BLOCK_SIZE,
    .mask_flags   = MTD_WRITEABLE,  /* force read-only */
  }
}
```

Any command-line input or output is written as follows:

```
# flash_erase -j /dev/mtd6 0 0
# nandwrite /dev/mtd6 rootfs-sum.jffs2
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "The second line prints the message **Please press Enter to activate this console** on the console."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail `feedback@packtpub.com`, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files from your account at `http://www.packtpub.com` for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support` and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.