Essential Linux Device Drivers

By Sreekrishnan Venkateswaran

Essential Linux Device Drivers By Sreekrishnan Venkateswaran

“Probably the most wide ranging and complete Linux device driver book I’ve read.”
--Alan Cox, Linux Guru and Key Kernel Developer

“Very comprehensive and detailed, covering almost every single Linux device driver type.”
--Theodore Ts’o, First Linux Kernel Developer in North America and Chief Platform Strategist of the Linux Foundation

The Most Practical Guide to Writing Linux Device Drivers
Linux now offers an exceptionally robust environment for driver development: with today’s kernels, what once required years of development time can be accomplished in days. In this practical, example-driven book, one of the world’s most experienced Linux driver developers systematically demonstrates how to develop reliable Linux drivers for virtually any device. ***Essential Linux Device Drivers*** is for any programmer with a working knowledge of operating systems and C, including programmers who have never written drivers before. Sreekrishnan Venkateswaran focuses on the essentials, bringing together all the concepts and techniques you need, while avoiding topics that only matter in highly specialized situations. Venkateswaran begins by reviewing the Linux 2.6 kernel capabilities that are most relevant to driver developers. He introduces simple device classes; then turns to serial buses such as I2C and SPI; external buses such as PCMCIA, PCI, and USB; video, audio, block, network, and wireless device drivers; user-space drivers; and drivers for embedded Linux–one of today’s fastest growing areas of Linux development. For each, Venkateswaran explains the technology, inspects relevant kernel source files, and walks through developing a complete example.

• Addresses drivers discussed in no other book, including drivers for I2C, video, sound, PCMCIA, and different types of flash memory
• Demystifies essential kernel services and facilities, including kernel threads and helper interfaces
• Teaches polling, asynchronous notification, and I/O control
• Introduces the Inter-Integrated Circuit Protocol for embedded Linux drivers
• Covers multimedia device drivers using the Linux-Video subsystem and Linux-Audio framework
• Shows how Linux implements support for wireless technologies such as Bluetooth, Infrared, WiFi, and cellular networking
• Describes the entire driver development lifecycle, through debugging and

maintenance
• Includes reference appendixes covering Linux assembly, BIOS calls, and Seq files

Essential Linux Device Drivers

By Sreekrishnan Venkateswaran

Essential Linux Device Drivers By Sreekrishnan Venkateswaran

“Probably the most wide ranging and complete Linux device driver book I’ve read.”
--Alan Cox, Linux Guru and Key Kernel Developer

“Very comprehensive and detailed, covering almost every single Linux device driver type.”
--Theodore Ts’o, First Linux Kernel Developer in North America and Chief Platform Strategist of the Linux Foundation

The Most Practical Guide to Writing Linux Device Drivers
Linux now offers an exceptionally robust environment for driver development: with today’s kernels, what once required years of development time can be accomplished in days. In this practical, example-driven book, one of the world’s most experienced Linux driver developers systematically demonstrates how to develop reliable Linux drivers for virtually any device. ***Essential Linux Device Drivers*** is for any programmer with a working knowledge of operating systems and C, including programmers who have never written drivers before. Sreekrishnan Venkateswaran focuses on the essentials, bringing together all the concepts and techniques you need, while avoiding topics that only matter in highly specialized situations. Venkateswaran begins by reviewing the Linux 2.6 kernel capabilities that are most relevant to driver developers. He introduces simple device classes; then turns to serial buses such as I2C and SPI; external buses such as PCMCIA, PCI, and USB; video, audio, block, network, and wireless device drivers; user-space drivers; and drivers for embedded Linux–one of today’s fastest growing areas of Linux development. For each, Venkateswaran explains the technology, inspects relevant kernel source files, and walks through developing a complete example.

• Addresses drivers discussed in no other book, including drivers for I2C, video, sound, PCMCIA, and different types of flash memory
• Demystifies essential kernel services and facilities, including kernel threads and helper interfaces
• Teaches polling, asynchronous notification, and I/O control
• Introduces the Inter-Integrated Circuit Protocol for embedded Linux drivers
• Covers multimedia device drivers using the Linux-Video subsystem and Linux-Audio framework
• Shows how Linux implements support for wireless technologies such as Bluetooth, Infrared, WiFi, and cellular networking
• Describes the entire driver development lifecycle, through debugging and maintenance
• Includes reference appendixes covering Linux assembly, BIOS calls, and Seq files

Essential Linux Device Drivers By Sreekrishnan Venkateswaran Bibliography

- Sales Rank: #469437 in Books
- Brand: Venkateswaran, Sreekrishnan
- Published on: 2008-04-06
- Original language: English
- Number of items: 1

- Dimensions: 9.50" h x 1.69" w x 7.34" l, 2.82 pounds
- Binding: Hardcover
- 744 pages

Essential Linux Device Drivers

by Sreekrishnan Venkiteswaran

Preface

It was the late nineties and at IBM, we were putting the Linux kernel on a wrist watch. The target device was tiny, but the task was turning out to be tough. The Memory Technology Devices subsystem didn't exist in the kernel, which meant that before a filesystem could start life on the watch's flash memory, we had to develop the necessary storage driver from scratch. Interfacing the watch's touch screen with user applications was complicated since the kernel's *input* event driver interface hadn't been conceived yet. Getting X-Windows to run on the watch's LCD wasn't easy since it didn't work well with framebuffer drivers. Of what use is a water-proof Linux wrist watch if you can't stream stock quotes from your bath tub? Bluetooth integration with Linux was several years away, and months were spent porting a proprietary Bluetooth stack to Internet-enable the watch. Power management support was good enough only to squeeze a few hours of juice from the watch's battery, hence we had work cut out on that front too. Linux-Infrared was still unstable, so we had to coax the stack before we could use an Infrared keyboard for data entry. And we had to compile the compiler and cross-compile a compact application-set since there were no accepted distributions in the consumer electronics space.

Fast forward to the present: The baby penguin has grown into a healthy teenager. What took thousands of lines of code and a year in development back then, can be accomplished in a few days with the current kernels. But to become a versatile kernel engineer who can magically weave solutions, you need to understand the myriad features and facilities that Linux offers today.

About the Book

Among the various subsystems residing in the kernel source tree, the ***drivers/*** directory constitutes the single largest chunk and is several times bigger than the others. With new and diverse technologies arriving in popular form factors, the development of new device drivers in the kernel is accelerating steadily. The latest kernels support over 50 device driver families.

This book is about writing Linux device drivers. It covers the design and development of major device classes supported by the kernel, including those I missed during my Linux-on-Watch days. The discussion of each driver family starts by looking at the corresponding technology, moves on to develop a practical example, and ends by looking at relevant kernel source files. But before foraying into the world of device drivers, the book introduces you to the kernel and discusses the important features of 2.6 Linux, emphasizing those portions that are of special interest to device driver writers.

Audience

This book is intended for the intermediate-level programmer eager to tweak the kernel to enable new devices. You should have a working knowledge of operating system concepts. For example, you should know what a system call is, and why concurrency issues have to be factored in while writing kernel code. The book assumes that you have downloaded Linux on your system, poked through the kernel sources, and at least skimmed through some related documentation. And you should be pretty good in C.

Summary of Chapters

The first three chapters prepare you to digest the rest of the book. Each of the next fifteen chapters discusses drivers for a specific device family. The following chapter is a hold-all for driver classes not covered thus far. The penultimate chapter discusses device driver debugging. The last chapter gives some perspective on delivery and maintenance.

Chapter 1, "Introduction," starts our tryst with Linux. It hurries you through downloading the kernel sources, making trivial code changes, and building a bootable kernel image.

Chapter 2, "A Peek Inside the Kernel," takes a brisk peek into the innards of the Linux kernel and teaches you some must-know kernel concepts. It first takes you through the boot process and then describes kernel services particularly relevant to driver development such as kernel threads, timers, concurrency, and memory management.

Chapter 3, "Getting Started with Device Drivers," gets you started with the art of writing Linux device drivers. It looks at interrupt handling, the new Linux device model, and Linux assembly. In this chapter, you'll also learn to use kernel helper interfaces such as linked lists, work queues, completion functions, and notifier chains. These helper facilities simplify your code, weed out redundancies from the kernel, and help long-term maintenance.

Chapter 4, "Character Drivers," looks at the architecture of character device drivers. Several concepts introduced in this chapter such as polling, asynchronous notification, and I/O control, are relevant to subsequent chapters as well, since many device classes discussed in the rest of the book are 'super' character devices.

Chapter 5, "Serial Drivers," explains the kernel layer that handles serial devices. The serial layer consists of low-level drivers, the TTY layer, and *line disciplines*.

Chapter 6, "Input Drivers," discusses the kernel's *input* subsystem that is responsible for servicing devices such as keyboards, mice, and touch panels.

Chapter 7, "The Inter-Integrated Circuit Protocol," dissects drivers for devices such as EEPROMs that are connected to the system I^2C bus or SMBus. The chapter also looks at other serial technologies such as the SPI bus and one-wire bus.

Chapter 8,"PCMCIA and Compact Flash," delves into the PCMCIA subsystem. It teaches you to write drivers for devices having a PCMCIA or Compact Flash form factor.

Chapter 9, "Peripheral Component Interconnect," looks at kernel support for PCI and its derivatives such as CardBus and PCI Express.

Chapter 10, "Universal Serial Bus," explores USB architecture and device drivers.

Chapter 11, "Video Drivers," explains the Linux video family.

Chapter 12, "Audio Drivers," describes the Linux audio family.

C*hapter 13, "Block Drivers,"* covers drivers for devices such as IDE and SCSI. It also looks at filesystem drivers.

Chapter 14, "Network Interface Cards," is dedicated to network devices. You'll learn about kernel networking data structures and how to interface network drivers with protocol layers.

Chapter 15, "Linux Without Wires," looks at driving different wireless technologies such as Bluetooth, Infrared, WiFi and cellular communication.

Chapter 16, "Memory Technology Devices," discusses flash memory enablement. This chapter first looks at flash-based protocols and chipsets primarily used on embedded devices. It ends by examining drivers for the Firmware Hub found on desktops and laptops.

Chapter 17, "Embedding Linux," steps into the world of embedded Linux. It takes you through the main firmware components of an embedded solution, such as bootloader, kernel, and device drivers. Given the soaring popularity of Linux in the embedded space, it's likely that you'll use the device driver skills that you acquire from this book, to enable embedded devices.

Chapter 18, "User Mode Drivers," looks at driving different types of devices from user space. Some device drivers, especially ones that are heavy on policy and light on performance requirements, are better off residing in user land. This chapter also explains how the new ultra-scalable process scheduler improves response times of user mode drivers.

Chapter 19, "More Devices and Drivers," takes a tour of a potpourri of driver families not covered thus far, such as Error Detection And Correction (EDAC), *cpufreq*, FireWire and ACPI.

Chapter 20, "Debugging Device Drivers," teaches about different types of debuggers that you can use to debug kernel code. In this chapter, you'll also learn to use trace tools, kernel probes, crash-dump, and profilers. When you develop a driver, be armed with the driver debugging skills that you learn in this chapter.

Chapter 21, "Delivery and Maintenance," provides perspective on the software life cycle and ponders *What next?*

The book is generally organized according to device and bus complexity, coupled with practical reasons of dependencies between chapters. So, we start off with basic device classes such as character, serial, and input. Next, we look at simple serial buses such as I^2C and SMBus. External buses such as PCMCIA, PCI and USB follow. Video, audio, block, and network devices usually interface with the system via I/O buses, so we look at them soon after. The next three chapters are oriented towards embedded Linux, and cover technologies such as wireless networking and flash memory. User space drivers are distilled out towards the end of the book.

Conventions Used

Source code, function names and shell commands are written `like this.` The shell prompt used is "**`bash>`** ". Filename are written in italics *like this*. Italics are also used to introduce new terms.

Some chapters modify original kernel source files while implementing code examples. To clearly point out the changes, newly inserted code lines are prefixed with '+', and any deleted code lines with '-'.

Sometimes, for simplicity, the book uses generic references. So if the text points you to the *arch/your-arch/* directory, it should be translated for example, to *arch/i386/* if you are compiling the kernel for the *x86* architecture. Similarly, any mention of the *include/asm-your-arch/* directory should be read as *include/asm-ppc/* if you are for instance, building the kernel for the POWER architecture. The '*' symbol and '*X*' are

occasionally used as a wild card character in filenames. So, if a chapter asks you to look at *include/linux/time*.h*, look at the header files, *time.h*, *timer.h*, *times.h* and *timex.h*, residing in the *include/linux/* directory. If a section talks about */dev/input/eventX* or */sys/devices/platform/i8042/serioX/*, *X* is the interface number assigned to your device in the context of your system configuration.

Simple regular expressions are occasionally used to compactly list function prototypes. For example, the section "*Direct Memory Access*" in *Chapter 9*, "*Peripheral Component Interconnect*", refers to `pci_mapunmapdma_sync_single()` instead of explicitly citing `pci_map_single()`, `pci_umap_single()`, and `pci_dma_sync_single()`.

Several chapters refer you to user space configuration files. For example, the section that describes the boot process opens */etc/rc.sysinit*, while the chapter on PCMCIA drivers flips through */etc/pcmcia/config*. The exact names and locations of such files might, however, vary according to the Linux distribution you use.

Dedication

This book is dedicated to the ten million visually impaired citizens of India. All author proceeds will go to their cause.

Read Essential Linux Device Drivers By Sreekrishnan Venkateswaran for online ebook

Essential Linux Device Drivers By Sreekrishnan Venkateswaran Free PDF d0wnl0ad, audio books, books to read, good books to read, cheap books, good books, online books, books online, book reviews epub, read books online, books to read online, online library, greatbooks to read, PDF best books to read, top books to read Essential Linux Device Drivers By Sreekrishnan Venkateswaran books to read online.

Online Essential Linux Device Drivers By Sreekrishnan Venkateswaran ebook PDF download

Essential Linux Device Drivers By Sreekrishnan Venkateswaran Doc

Essential Linux Device Drivers By Sreekrishnan Venkateswaran Mobipocket

Essential Linux Device Drivers By Sreekrishnan Venkateswaran EPub