Linux Kernel Development Acquisitions Editor

Linux Device Driver Development
Linux Kernel Development
Linux Kernel Debugging
Professional Linux Programming
Linux Kernel Programming
Linux Kernel Programming
Linux Kernel in a Nutshell
Linux Kernel Development
Mastering Linux Kernel Development
The Linux Kernel Primer
Linux Kernel Programming
Embedded Linux System Development
Linux Kernel Development
Linux for Embedded and Real-time Applications
Linux Internals
The Linux Kernel Module Programming Guide
The Art of Linux Kernel Design
Advanced Linux Programming
Professional Linux Kernel Architecture
Linux System Programming
Understanding the Linux Kernel
HTML5
Professional Linux Kernel Programming
Understanding the Linux Kernel
Embedded Linux Development Using Eclipse
Beginning Linux?Programming
Linux Kernel and Driver Development: Training Handouts
Embedded Linux Development Using Yocto Project Cookbook
Linux Kernel
Linux Kernel Development
Linux Kernel Guide Book
Linux Kernel Programming
Linux Kernel and Driver Development - Practical Labs
Linux Kernel Programming Part 2 - Char Device Drivers and Kernel Synchronization
The Linux Programming Interface
User Mode Linux
Mastering Linux Kernel Development
Linux Kernel and Device Driver Programming
Exploring Raspberry Pi

DEMARION CASSIDY

Linux Device Driver Development Packt Publishing Ltd
An authoritative, practical guide that helps programmers better understand the Linux kernel and to write and develop kernel code.
Linux Kernel Development Createspace Independent Publishing Platform
Linux Kernel Development details the design and implementation of the Linux kernel, presenting the content in a manner that is beneficial to those writing and developing kernel code, as well as to programmers seeking to better understand the operating system and become more efficient and productive in their coding. The book details the major subsystems and features of the Linux kernel, including its design, implementation, and interfaces. It covers the Linux kernel with both a practical and theoretical eye, which should appeal to readers with a variety of interests and needs. The author, a core kernel developer, shares valuable knowledge and experience on the 2.6 Linux kernel. Specific topics covered include process management, scheduling, time management and timers, the system call interface, memory addressing, memory management, the page cache, the VFS, kernel synchronization, portability concerns, and debugging techniques. This book covers the most interesting features of the Linux 2.6 kernel, including the CFS scheduler, preemptive kernel, block I/O layer, and I/O schedulers. The third edition of Linux Kernel Development includes new and updated material throughout the book: An all-new chapter on kernel data structures Details on interrupt handlers and bottom halves Extended coverage of virtual memory and memory allocation Tips on debugging the Linux kernel In-depth coverage of kernel synchronization and locking Useful insight into submitting kernel patches and working with the Linux kernel community
Linux Kernel Debugging John Wiley & Sons
This is the eBook version of the printed book. If the print book includes a CD-ROM, this content is not included within the eBook version. Advanced Linux Programming is divided into two parts. The first covers generic UNIX system services, but with a particular eye towards Linux specific information. This portion of the book will be of use even to advanced programmers who have worked with other Linux systems since it will cover Linux specific details and differences. For programmers without UNIX experience, it will be even more valuable. The second section covers material that is entirely Linux specific. These are truly advanced topics, and are the techniques that the gurus use to build great applications. While this book will focus mostly on the Application Programming Interface (API) provided by the Linux kernel and the C library, a preliminary introduction to the development tools available will allow all who purchase the book to make immediate use of Linux.
Professional Linux Programming Packt Publishing Ltd
Presents an overview of kernel configuration and building for version 2.6 of the Linux kernel.
Linux Kernel Programming Linux Kernel Development
Expand Raspberry Pi capabilities with fundamental engineering principles Exploring Raspberry Pi is the innovators guide to bringing Raspberry Pi to life. This book favors engineering principles over a 'recipe' approach to give you the skills you need to design and build your own projects. You'll understand the fundamental principles in a way that transfers to any type of electronics, electronic modules, or external peripherals, using a "learning by doing" approach that caters to both beginners and experts. The book begins with basic Linux and programming skills, and helps you stock your inventory with common parts and supplies. Next, you'll learn how to make parts work together to achieve the goals of your project, no matter what type of components you use. The companion website provides a full repository that structures all of the code and scripts, along with links to video tutorials and supplementary content that takes you deeper into your project. The Raspberry Pi's most famous feature is its adaptability. It can be used for thousands of electronic applications, and using the Linux OS expands the functionality even more. This book helps you get the most from your Raspberry Pi, but it also gives you the fundamental engineering skills you need to incorporate any electronics into any project. Develop the Linux and programming skills you need to build basic applications Build your inventory of parts so you can always "make it work" Understand interfacing, controlling, and communicating with almost any component Explore advanced applications with video, audio, real-world interactions, and more Be free to adapt and create with Exploring Raspberry Pi.
Linux Kernel Programming Prentice-Hall PTR
Linux Kernel Development, Second Edition details the design and implementation of the Linux kernel, presenting the content in a manner that is beneficial to those writing and developing kernel code. While the book discusses topics that are theoretical, it does so with the goal of assisting programmers so they better understand the topics and become more efficient and productive in their coding. The book discusses the major subsystems and features of the Linux kernel, including design and implementation, their purpose and goals, and their interfaces. Important computer science and operating system design details are also addressed. The book covers the Linux kernel from both angles--theoretical and applied--which should appeal to both types of readers. The author is involved in Linux kernel development, so the latest kernel version is detailed, as the author has access to the not-yet-released development releases.; Specific topics covered will include all the important algorithms, relevant subsystems, process management, scheduling, time management and timers, system call interface, memory addressing, memory management, paging strategies, caching layers, VFS, kernel synchronization, and signals.
Linux Kernel in a Nutshell "O'Reilly Media, Inc."
This book contains the practical labs corresponding to the "Linux Kernel and Driver Development: Training Handouts" book from Bootlin. Get your hands on an embedded board based on an ARM processor (the Beagle Bone Black board), and apply what you learned: write a Device Tree to declare devices connected to your board, configure pin multiplexing, and implement drivers for I2C and serial devices. You will learn how to manage multiple devices with the same driver, to acces and write hardware registers, to allocate memory, to register and manage interrupts, as well as how to debug your code and interpret the kernel error messages. You will also keep an eye on the board and CPU datasheets so that you will always understand the values that you feed to the kernel.
Linux Kernel Development Prentice-Hall PTR
Effectively debug kernel modules, device drivers, and the kernel itself by gaining a solid understanding of powerful open source tools and advanced kernel debugging techniques Key Features • Fully understand how to use a variety of kernel and module debugging tools and techniques using examples • Learn to expertly interpret a kernel Oops and identify underlying defect(s) • Use easy-to-look up tables and clear explanations of kernel-level defects to make this complex topic easy Book Description The Linux kernel is at the very core of arguably the world's best production-quality OS. Debugging it, though, can be a complex endeavor. Linux Kernel Debugging is a comprehensive guide to learning all about advanced kernel debugging. This book covers many areas in-depth, such as instrumentation-based debugging techniques (printk and the dynamic debug framework), and shows you how to use Kprobes. Memory-related bugs tend to be a nightmare – two chapters are packed with tools and techniques devoted to debugging them. When the kernel gifts you an Oops, how exactly do you interpret it to be able to debug the underlying issue? We've got you covered. Concurrency tends to be an inherently complex topic, so a chapter on lock debugging will help you to learn precisely what data races are, including using KCSAN to detect them. Some thorny issues, both debug- and performance-wise, require detailed kernel-level tracing; you'll learn to wield the impressive power of Ftrace and its frontends. You'll also discover how to handle kernel lockups, hangs, and the dreaded kernel panic, as well as leverage the venerable GDB tool within the kernel (KGDB), along with much more. By the end of this book, you will have at your disposal a wide range of powerful kernel debugging tools and techniques, along with a keen sense of

when to use which. What you will learn • Explore instrumentation-based printk along with the powerful dynamic debug framework • Use static and dynamic Kprobes to trap into kernel/module functions • Catch kernel memory defects with KASAN, UBSAN, SLUB debug, and kmemleak • Interpret an Oops in depth and precisely identify it's source location • Understand data races and use KCSAN to catch evasive concurrency defects • Leverage Ftrace and trace-cmd to trace the kernel flow in great detail • Write a custom kernel panic handler and detect kernel lockups and hangs • Use KGDB to single-step and debug kernel/module source code Who this book is for This book is for Linux kernel developers, module/driver authors, and testers interested in debugging and enhancing their Linux systems at the level of the kernel. System administrators who want to understand and debug the internal infrastructure of their Linux kernels will also find this book useful. A good grasp on C programming and the Linux command line is necessary. Some experience with kernel (module) development will help you follow along.

Sams Publishing

This book contains the practical labs corresponding to the "Embedded Linux System Development: Training Handouts" book from Bootlin. Get your hands on an embedded board based on an ARM processor (the Atmel/Microchip SAMA5D3 Xplained board), and apply what you learned to: make you own cross-compiling toolchain, compile and install your bootloader and Linux kernel, make a custom root filesystem, manage your storage in an efficient and reliable way, cross-compile extra open-source component together with your own applications, implement real-time requirements so that you can quickly turn your ideas into a working prototype!

Mastering Linux Kernel Development "O'Reilly Media, Inc."

Explore Implementation of core kernel subsystems About This Book Master the design, components, and structures of core kernel subsystems Explore kernel programming interfaces and related algorithms under the hood Completely updated material for the 4.12.10 kernel Who This Book Is For If you are a kernel programmer with a knowledge of kernel APIs and are looking to build a comprehensive understanding, and eager to explore the implementation, of kernel subsystems, this book is for you. It sets out to unravel the underlying details of kernel APIs and data structures, piercing through the complex kernel layers and gives you the edge you need to take your skills to the next level. What You Will Learn Comprehend processes and fles—the core abstraction mechanisms of the Linux kernel that promote effective simplification and dynamism Decipher process scheduling and understand effective capacity utilization under general and real-time dispositions Simplify and learn more about process communication techniques through signals and IPC mechanisms Capture the rudiments of memory by grasping the key concepts and principles of physical and virtual memory management Take a sharp and precise look at all the key aspects of interrupt management and the clock subsystem Understand concurrent execution on SMP platforms through kernel synchronization and locking techniques In Detail Mastering Linux Kernel Development looks at the Linux kernel, its internal arrangement and design, and various core subsystems, helping you to gain significant understanding of this open source marvel. You will look at how the Linux kernel, which possesses a kind of collective intelligence thanks to its scores of contributors, remains so elegant owing to its great design. This book also looks at all the key kernel code, core data structures, functions, and macros, giving you a comprehensive foundation of the implementation details of the kernel's core services and mechanisms. You will also look at the Linux kernel as well-designed software, which gives us insights into software design in general that are easily scalable yet fundamentally strong and safe. By the end of this book, you will have considerable understanding of and appreciation for the Linux kernel. Style and approach Each chapter begins with the basic conceptual know-how for a subsystem and extends into the details of its implementation. We use appropriate code excerpts of critical routines and data structures for subsystems.

The Linux Kernel Primer Packt Publishing Ltd

Using the training lecture materials from Bootlin, learn how to make the Linux kernel support new hardware, both for driving new devices and for supporting a new board. You will get familiar with how Linux abstracts the hardware and how it uses buses to bind devices to drivers. This book also covers the infrastructure that Linux offers to support device driver development: managing memory, mapping registers, registering interrupt handlers, locking and debugging primitives. To run the practical labs, you will need an affordable electronic board, and the corresponding - "Training Labs" booklet.

Linux Kernel Programming Createspace Independent Publishing Platform

Over 79 hands-on recipes for professional embedded Linux developers to optimize and boost their Yocto Project know-how Key Features Optimize your Yocto setup to speed up development and debug build issues Use what is quickly becoming the standard embedded Linux product builder framework—the Yocto Project Recipe-based implementation of best practices to optimize your Linux system Book Description The Yocto Project has become the de facto distribution build framework for reliable and robust embedded systems with a reduced time to market.You'll get started by working on a build system where you set up Yocto, create a build directory, and learn how to debug it. Then, you'll explore everything about the BSP layer, from creating a custom layer to debugging device tree issues. In addition to this, you'll learn how to add a new software layer, packages, data, scripts, and configuration files to your system. You will then cover topics based on application development, such as using the Software Development Kit and how to use the Yocto project in various development environments. Toward the end, you will learn how to debug, trace, and profile a running system. This second edition has been updated to include new content based on the latest Yocto release. What you will learn Optimize your Yocto Project setup to speed up development and debug build issues Use Docker containers to build Yocto Project-based systems Take advantage of the user-friendly Toaster web interface to the Yocto Project build system Build and debug the Linux kernel and its device trees Customize your root filesystem with already-supported and new Yocto packages Optimize your production systems by reducing the size of both the Linux kernel and root filesystems Explore the mechanisms to increase the root filesystem security Understand the open source licensing requirements and how to comply with them when cohabiting with proprietary programs Create recipes, and build and run applications in C, C++, Python, Node.js, and Java Who this book is for If you are an embedded Linux developer with the basic knowledge of Yocto Project, this book is an ideal way to broaden your knowledge with recipes for embedded development.

Embedded Linux System Development "O'Reilly Media, Inc."

This book is an exploration of the Linux Kernel. The first part of the book is a guide for you on how to work with the Initial RAM Disk (initrd). This simply provides us with an easy way to load the RAM disk using the boot loader. The necessary steps which can help you achieve this, and the necessary tools for you have been discussed. The tools which can be used for kernel development are discussed in this book. The first tool discussed in this book is the kcov. The book guides you on how to get started with this tool for the purpose of kernel development to the final stages. The coccinelle, which is a tool for kernel development is also examined in this book. This is a good tool which can help you in pattern matching and in the transformation of text. You are guided on how to install this tool and then how to use it for the purpose of kernel development. Lastly, the book guides you on how to write or create the Linux kernel modules. This means that you will learn how to create modules for the Linux kernel on your own. The following topics are discussed in this book: - Initial RAM Disk (initrd) - Kernel Development Tools - Writing Linux Kernel Modules

Linux Kernel Development CRC Press

Discover how to write high-quality character driver code, interface with userspace, work with chip memory, and gain an in-depth understanding of working with hardware interrupts and kernel synchronization Key FeaturesDelve into hardware interrupt handling, threaded IRQs, tasklets, softirqs, and understand which to use whenExplore powerful techniques to perform user-kernel interfacing, peripheral I/O and use kernel mechanismsWork with key kernel synchronization primitives to solve kernel concurrency issuesBook Description Linux Kernel Programming Part 2 - Char Device Drivers and Kernel Synchronization is an ideal companion guide to the Linux Kernel Programming book. This book provides a comprehensive introduction for those new to Linux device driver development and will have you up and running with writing misc class character device driver code (on the 5.4 LTS Linux kernel) in next to no time. You'll begin by learning how to write a simple and complete misc class character driver before interfacing your driver with user-mode processes via procfs, sysfs, debugfs, netlink sockets, and ioctl. You'll then find out how to work with hardware I/O memory. The book covers working with hardware interrupts in depth and helps you understand interrupt request (IRQ) allocation, threaded IRQ handlers, tasklets, and softirqs. You'll also explore the practical usage of useful kernel mechanisms, setting up delays, timers, kernel threads, and workqueues. Finally, you'll discover how to deal with the complexity of kernel synchronization with locking technologies (mutexes, spinlocks, and atomic/refcount operators), including more advanced topics such as cache effects, a primer on lock-free techniques, deadlock avoidance (with lockdep), and kernel lock debugging techniques. By the end of this Linux kernel book, you'll have learned the fundamentals of writing Linux character device driver code for real-world projects and products. What you will learnGet to grips with the basics of the modern Linux Device Model (LDM)Write a simple yet complete misc class character device driverPerform user-kernel interfacing using popular methodsUnderstand and handle hardware interrupts confidentlyPerform I/O on peripheral hardware chip memoryExplore kernel APIs to work with delays, timers, kthreads, and workqueuesUnderstand kernel concurrency issuesWork with key kernel synchronization primitives and discover how to detect and avoid deadlockWho this book is for An understanding of the topics covered in the Linux Kernel Programming book is highly recommended to make the most of this book. This book is for Linux programmers beginning to find their way with device driver development. Linux device driver developers looking to overcome frequent and common kernel/driver development issues, as well as perform common driver tasks such as user-kernel interfaces, performing peripheral I/O, handling hardware interrupts, and dealing with concurrency will benefit from this book. A basic understanding of Linux kernel internals (and common APIs), kernel module development, and C programming is required.

Linux for Embedded and Real-time Applications Packt Publishing Ltd

Write software that draws directly on services offered by the Linux kernel and core system libraries. With this comprehensive book, Linux kernel contributor Robert Love provides you with a tutorial on Linux system programming, a reference manual on Linux system calls, and an insider’s guide to writing smarter, faster code. Love clearly distinguishes between POSIX standard functions and special services offered only by Linux. With a new chapter on multithreading, this updated and expanded edition provides an in-depth look at Linux from both a theoretical and applied perspective over a wide range of programming topics, including: A Linux kernel, C library, and C compiler overview Basic I/O operations, such as reading from and writing to files Advanced I/O interfaces, memory mappings, and optimization techniques The family of system calls for basic process management Advanced process management, including real-time processes Thread concepts, multithreaded programming, and Pthreads File and directory management Interfaces for allocating memory and optimizing memory access Basic and advanced signal interfaces, and their role on the system Clock management, including POSIX clocks and high-resolution timers

Linux Internals CreateSpace

Furnishing in-depth coverage of Linux source-code internals, this high-level handbook explains how the Linux system operating system works and how to use it with various programming applications, discussing the various Linux versions, performance and tuning issues, kernel programming, troubleshooting details, and other important topics. Original. (Intermediate)

The Linux Kernel Module Programming Guide John Wiley & Sons

With User Mode Linux you can create virtual Linux machines within a Linux computer and use them to test and debug applications, network services, and even kernels. This work covers almost everything from getting started through running enterprise-class User Mode Linux servers. It offers advice on bootup, compilation, administration, and more.

The Art of Linux Kernel Design Createspace Independent Publishing Platform

Use the latest version of HTML to create dynamic Web pages HTML5 is the latest iteration of the standard markup language for creating Web pages. It boasts extensive updates from its predecessor and allows you to incorporate rich media content into a site without any dependence on extra software such as Flash. Packed with hundreds of screen shots, this visual guide introduces you to the many new features and abilities of HTML5 and shows you the many exciting new possibilities that exist for designing dynamic Web pages. Offers visual learners a solid reference on HTML5, the latest version of the standard markup language for designing Web pages Demonstrates how to use HTML5 to create Web pages that feature the latest in rich media content Provides easy-to-understand examples that cover a variety of topics to get you up and running with HTML5 Features a companion Web site that contains all the code needed to learn HTML5 HTML5: Your visual blueprint for designing effective Web pages opens your eyes to the world of possibilities that exist with the new version of the popular markup language. Adam R. McDaniel is a Web developer, technical architect, and security analyst, who has contributed to the Linux Kernel.

Advanced Linux Programming Createspace Independent Publishing Platform

Uses the Running Operation as the Main Thread Difficulty in understanding an operating system (OS) lies not in the technical aspects, but in the complex relationships inside the operating systems. The Art of Linux Kernel Design: Illustrating the Operating System Design Principle and Implementation addresses this complexity. Written from the perspective of the designer of an operating system, this book tackles important issues and practical problems on how to understand an operating system completely and systematically. It removes the mystery, revealing operating system design guidelines, explaining the BIOS code directly related to the operating system, and simplifying the relationships and guiding ideology behind it all. Based on the Source Code of a Real Multi-Process Operating System Using the 0.11 edition source code as a representation of the Linux basic design, the book illustrates the real states of an operating system in actual operations. It provides a complete, systematic analysis of the operating system source code, as well as a direct and complete understanding of the real operating system run-time structure. The author includes run-time memory structure diagrams, and an accompanying essay to help readers grasp the dynamics behind Linux and similar software systems. Identifies through diagrams the location of the key operating system data structures that lie in the memory Indicates through diagrams the current operating status information which helps users understand the interrupt state, and left time slice of processes Examines the relationship between process and memory, memory and file, file and process, and the kernel Explores the essential association, preparation, and transition, which is the

vital part of operating system Develop a System of Your Own This text offers an in-depth study on mastering the operating system, and provides an important prerequisite for designing a whole new operating system.

Professional Linux Kernel Architecture Computing McGraw-Hill

Learn how to write high-quality kernel module code, solve common Linux kernel programming issues, and understand the fundamentals of Linux kernel internals Key Features Discover how to write kernel code using the Loadable Kernel Module framework Explore industry-grade techniques to perform efficient memory allocation and data synchronization within the kernel Understand the essentials of key internals topics such as kernel architecture, memory management, CPU scheduling, and kernel synchronization Book DescriptionLinux Kernel Programming is a comprehensive introduction for those new to Linux kernel and module development. This easy-to-follow guide will have you up and running with writing kernel code in next-to-no time. This book uses the latest 5.4 Long-Term Support (LTS) Linux kernel, which will be maintained from November 2019 through to December 2025. By working with the 5.4 LTS kernel throughout the book, you can be confident that your knowledge will continue to be valid for years to come. You'll start the journey by learning how to build the kernel from the source. Next, you'll write your first kernel module using the powerful Loadable Kernel Module (LKM) framework. The following chapters will cover key kernel internals topics including Linux kernel architecture, memory management, and CPU scheduling. During the course of this book, you'll delve into the fairly complex topic of concurrency within the kernel, understand the issues it can cause, and learn how they can be addressed with various locking technologies (mutexes, spinlocks, atomic, and refcount operators). You'll also benefit from more advanced material on cache effects, a primer on lock-free techniques within the kernel, deadlock avoidance (with lockdep), and kernel lock debugging techniques. By the end of this kernel book, you'll have a detailed understanding of the fundamentals of writing Linux kernel module code for real-world projects and products.What you will learn Write high-quality modular kernel code (LKM framework) for 5.x kernels Configure and build a kernel from source Explore the Linux kernel architecture Get to grips with key internals regarding memory management within the kernel Understand and work with various dynamic kernel memory alloc/dealloc APIs Discover key internals aspects regarding CPU scheduling within the kernel Gain an understanding of kernel concurrency issues Find out how to work with key kernel synchronization primitives Who this book is for This book is for Linux programmers beginning to find their way with Linux kernel development. If you're a Linux kernel and driver developer looking to overcome frequent and common kernel development issues, or understand kernel intervals, you'll find plenty of useful information. You'll need a solid foundation of Linux CLI and C programming before you can jump in.