

Linux Pci Device Driver A Template Linux Driver Development

Linux Kernel Networking takes you on a guided in-depth tour of the current Linux networking implementation and the theory behind it. Linux kernel networking is a complex topic, so the book won't burden you with topics not directly related to networking. This book will also not overload you with cumbersome line-by-line code walkthroughs not directly related to what you're searching for; you'll find just what you need, with in-depth explanations in each chapter and a quick reference at the end of each chapter. Linux Kernel Networking is the only up-to-date reference guide to understanding how networking is implemented, and it will be indispensable in years to come since so many devices now use Linux or operating systems based on Linux, like Android, and since Linux is so prevalent in the data center arena, including Linux-based virtualization technologies like Xen and KVM.

••PCI EXPRESS is considered to be the most general purpose bus so it should appeal to a wide audience in this arena. •Today's buses are becoming more specialized to meet the needs of the particular system applications, building the need for this book. •Mindshare and their only competitor in this space, Solari, team up in this new book.

Linux is a fast-growing operating system with power and appeal, and enterprises worldwide are quickly adopting the system to utilize its benefits. But as with all operating systems, performance problems do occur causing system administrators to scramble into action. Finally, there is a complete reference for troubleshooting Linux—quickly! Linux Troubleshooting for System Administrators and Power Users is THE book for locating and solving problems and maintaining high performance in Red Hat® Linux and Novell® SUSE® Linux systems. This book not only teaches you how to troubleshoot Linux, it shows you how the system works—so you can attack any problem at its root. Should you reinstall if Linux does not boot? Or can you save time by troubleshooting the problem? Can you enhance performance when Linux hangs or runs slowly? Can you overcome problems with printing or accessing a network? This book provides easy-to-follow examples and an extensive look at the tools, commands, and scripts that make Linux run properly. A troubleshooting guide for all Linux users: Focuses on common problems with start-up, printing, login, the network, security, and more Restore Linux when boot, startup, or shutdown fails—and reinstall Linux properly when all troubleshooting fails Explains how to use some of the most popular Linux performance tools, including top, sar, vmstat, iostat, and free Handle storage problems and CPU slamming to ensure high Linux performance Solve hardware device problems by deciphering error messages and using the lspci tool Use backup/recover commands and tape libraries to create proper backups Identify and correct remote and network printing problems using spooler commands Gone are the days of searching online for solutions that are out of date and unreliable. Whether you are a system admin, developer, or user, this book is an invaluable resource for ensuring that Linux runs smoothly, efficiently, and securely.

CD-ROM contains: Linux kernel version 2.4.4, plus sources from other programs and documents from the Linux

Documentation Project.

Building Embedded Linux Systems

Advanced Industrial Control Technology

Linux Kernel Programming Part 2 - Char Device Drivers and Kernel Synchronization

Industrial Control Technology

A Simpler Approach to Linux Kernel

Beyond BIOS

Benvenuti describes the relationship between the Internet's TCP/IP implementation and the Linux Kernel so that programmers and advanced administrators can modify and fine-tune their network environment.

Learn how to write high-quality kernel module code, solve common Linux kernel programming issues, and understand the fundamentals of Linux kernel internals Key FeaturesDiscover how to write kernel code using the Loadable Kernel Module frameworkExplore industry-grade techniques to perform efficient memory allocation and data synchronization within the kernelUnderstand the essentials of key internals topics such as kernel architecture, memory management, CPU scheduling, and kernel synchronizationBook Description Linux 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 learnWrite high-quality modular kernel code (LKM framework) for 5.x kernelsConfigure and build a kernel from sourceExplore the Linux kernel architectureGet to grips with key internals regarding memory management within the kernelUnderstand and work with various dynamic kernel memory alloc/dealloc APIsDiscover key internals aspects regarding CPU scheduling within the kernelGain an understanding of kernel concurrency issuesFind out how to work with key kernel synchronization primitivesWho 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 internals, you'll find plenty of useful information. You'll need a solid foundation of Linux CLI and C programming before you can jump in.

Device drivers literally drive everything you're interested in--disks, monitors, keyboards, modems--everything outside the computer chip and memory. And writing device drivers is one of the few areas of programming for the Linux operating system that calls for unique, Linux-specific knowledge. For years now, programmers have relied on the classic Linux Device Drivers from O'Reilly to master this critical subject. Now in its third edition, this bestselling guide provides all the information you'll need to write drivers for a wide range of devices. Over the years the book has helped countless programmers learn: how to support computer peripherals under the Linux operating system how to develop and write software for new hardware under Linux the basics of Linux operation even if they are not expecting to write a driver The new edition of Linux Device Drivers is better than ever. The book covers all the significant changes to Version 2.6 of the Linux kernel, which simplifies many activities, and contains subtle new features that can make a driver both more efficient and more flexible. Readers will find new chapters on important types of drivers not covered previously, such as consoles, USB drivers, and more. Best of all, you don't have to be a kernel hacker to understand and enjoy this book. All you need is an understanding of the C programming language and some background in Unix system calls. And for maximum ease-of-use, the book uses full-featured examples that you can compile and run without special hardware. Today Linux holds fast as the most rapidly growing segment of the computer market and continues to win over enthusiastic adherents in many application areas. With this increasing support, Linux is now absolutely mainstream, and viewed as a solid platform for embedded systems. If you're writing device drivers, you'll want this book. In fact, you'll wonder how drivers are ever written without it.

Control engineering seeks to understand physical systems, using mathematical modeling, in terms of inputs, outputs and various components with different behaviors. It has an essential role in a wide range of control systems, from household appliances to space flight. This book provides an in-depth view of the technologies that are implemented in most varieties of modern industrial control engineering. A solid grounding is provided in traditional control techniques, followed by detailed examination of modern control techniques such as real-time, distributed, robotic, embedded, computer and wireless control technologies. For each technology, the book discusses its full profile, from the field layer and the control layer to the operator layer. It also includes all the interfaces in industrial control systems: between controllers and systems; between different layers; and between operators and systems. It not only describes the details of both real-time operating systems and distributed operating systems, but also provides coverage of the microprocessor boot code, which other books lack. In addition to working principles and operation mechanisms, this book emphasizes the practical issues of components, devices and hardware circuits, giving the specification parameters, install procedures, calibration and configuration methodologies needed for engineers to put the theory into practice. Documents all the key technologies

of a wide range of industrial control systems Emphasizes practical application and methods alongside theory and principles An ideal reference for practicing engineers needing to further their understanding of the latest industrial control concepts and techniques

Understanding the Linux Kernel

FreeBSD Device Drivers

Linux Troubleshooting for System Administrators and Power Users

Linux Kernel Networking

Implementation and Theory

Develop customized drivers for embedded Linux

“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

Get up to speed with the most important concepts in driver development and focus on common embedded system

requirements such as memory management, interrupt management, and locking mechanisms Key FeaturesWrite feature-rich and customized Linux device drivers for any character, SPI, and I2C deviceDevelop a deep understanding of locking primitives, IRQ management, memory management, DMA, and so onGain practical experience in the embedded side of Linux using GPIO, IIO, and input subsystemsBook Description Linux is by far the most-used kernel on embedded systems. Thanks to its subsystems, the Linux kernel supports almost all of the application fields in the industrial world. This updated second edition of Linux Device Driver Development is a comprehensive introduction to the Linux kernel world and the different subsystems that it is made of, and will be useful for embedded developers from any discipline. You'll learn how to configure, tailor, and build the Linux kernel. Filled with real-world examples, the book covers each of the most-used subsystems in the embedded domains such as GPIO, direct memory access, interrupt management, and I2C/SPI device drivers. This book will show you how Linux abstracts each device from a hardware point of view and how a device is bound to its driver(s). You'll also see how interrupts are propagated in the system as the book covers the interrupt processing mechanisms in-depth and describes every kernel structure and API involved. This new edition also addresses how not to write device drivers using user space libraries for GPIO clients, I2C, and SPI drivers. By the end of this Linux book, you'll be able to write device drivers for most of the embedded devices out there. What you will learnDownload, configure, build, and tailor the Linux kernelDescribe the hardware using a device treeWrite feature-rich platform drivers and leverage I2C and SPI busesGet the most out of the new concurrency managed workqueue infrastructureUnderstand the Linux kernel timekeeping mechanism and use time-related APIsUse the regmap framework to factor the code and make it genericOffload CPU for memory copies using DMAInteract with the real world using GPIO, IIO, and input subsystemsWho this book is for This Linux OS book is for embedded system and embedded Linux enthusiasts/developers who want to get started with Linux kernel development and leverage its subsystems. Electronic hackers and hobbyists interested in Linux kernel development as well as anyone looking to interact with the platform using GPIO, IIO, and input subsystems will also find this book useful.

Provides information on writing a driver in Linux, covering such topics as character devices, network interfaces, driver debugging, concurrency, and interrupts.

Eight minibooks in one volume cover every important aspect of Linux and everything you need to know to pass level-1 certification Linux All-in-One For Dummies explains everything you need to get up and running with the popular Linux operating system. Written in the friendly and accessible For Dummies style, the book ideal for new and intermediate Linux users, as well as anyone studying for level-1 Linux certification. The eight minibooks inside cover the basics of Linux, interacting with it, networking issues, Internet services, administration, security, scripting, and level-1 certification. Covers every major topic for anyone just getting familiar with Linux Includes a test-prep section for passing the level-1 Linux certification exam Written by the expert author of more than thirty books, including CompTIA Security+

Study Guide, 3rd Edition Including everything beginners need to know to get started with Linux, Linux All-in-One For Dummies, 5th Edition is the ultimate resource and reference for aspiring professionals.

Essential Linux Device Drivers

First Step Towards Device Driver Programming

PCI Express System Architecture

Understanding Linux Network Internals

Linux Kernel Programming

Linux Device Drivers Development

Discussions about server sprawl, rising software costs, going green, or moving data centers to reduce the cost of business are held in many meetings or conference calls in many organizations throughout the world. And many organizations are starting to turn toward System z™ and z/VM® after such discussions. The virtual machine operating system has over 40 years of experience as a hosting platform for servers, from the days of VM/SP, VM/XA, VM/ESA® and especially now with z/VM. With the consolidation of servers and conservative estimates that approximately seventy percent of all critical corporate data reside on System z, we find ourselves needing a highly secure environment for the support of this infrastructure. This document was written to assist z/VM support and security personnel in providing the enterprise with a safe, secure and manageable environment. This IBM® Redbooks® publication provides an overview of security and integrity provided by z/VM and the processes for the implementation and configuration of z/VM Security Server, z/VM LDAP Server, IBM Tivoli® Directory Server for z/OS®, and Linux® on System z with PAM for LDAP authentication. Sample scenarios with RACF® database sharing between z/VM and z/OS, or through Tivoli Directory Integrator to synchronize LDAP databases, are also discussed in this book. This book provides information about configuration and usage of Linux on System z with the System z Cryptographic features documenting their hardware and software configuration. The Consul zSecure Pro Suite is also part of this document: this product helps to control and audit security not only on one system, but can be used as a single point of enterprise wide security control. This document covers the installation and configuration of this product and detailed information is presented on

how z/Consul can be used to collect and analyze z/VM security data and how it can be helpful in the administration of your audit data.

Easy Linux Device Driver : "First Step Towards Device Driver Programming" Easy Linux Device Driver book is an easy and friendly way of learning device driver programming . Book contains all latest programs along with output screen screenshots. Highlighting important sections and stepwise approach helps for quick understanding of programming . Book contains Linux installation ,Hello world program up to USB 3.0 ,Display Driver ,PCI device driver programming concepts in stepwise approach. Program gives best understanding of theoretical and practical fundamentals of Linux device driver. Beginners should start learning Linux device driver from this book to become device driver expertise.--Topics Covered in book--*Introduction of LinuxAdvantages of Linux History of LinuxArchitecture of LinuxDefinitions*Ubuntu installationUbuntu Installation StepsUser Interface DifferenceAbout KNOPPIXImportant links*Terminal: Soul of LinuxCreating Root accountTerminal CommandsVirtual Editor Commands*Linux KernelLinux Kernel InternalsKernel Space and User space*Device DriverPlace of Driver in SystemDevice Driver working*Characteristics of Device Driver Module CommandsHello World Programpre-settingsWrite ProgramPrintk functionMakefileRun program*Parameter passingParameter passing programParameter Array*Process related program*Character Device DriverMajor and Minor numberAPI to registers a deviceProgram to show device numberCharacter Driver File OperationsFile operation program.Include .h headerFunctions in module.h fileImportant code snippetsSummary of file operations*PCI Device DriverDirect Memory AccessModule Device TableCode for Basic Device DriverImportant code snippets*USB Device Driver FundamentalsArchitecture of USB device driverUSB Device Driver programStructure of USB Device DriverParts of USB end pointsImportant featuresUSB information Driver*USB device Driver File OperationsUsing URBSimple data transferProgram to read and writelImportant code snippetsGadget Driver*Complete USB Device Driver ProgramSkeleton Driver Program*Special USB 3.0USB 3.0 Port connectionBulk endpoint streamingStream ID*Device Driver LockMutual ExclusionSemaphoreSpin Lock*Display Device DriverFrame buffer conceptFramebuffer Data StructureCheck and set ParameterAccelerated MethodDisplay Driver

summary*Memory AllocationKmallocVmallocioremap*Interrupt Handlinginterrupt registrationProc interfacePath of interruptProgramming TipsSoftirqs, Tasklets, Work Queues*I/O ControlIntroducing ioctlPrototypeStepwise execution of ioctl*Sample Device Driver Complete memory DriverComplete Parallel Port Driver*Device Driver DebuggingData Display DebuggerGraphical Display DebuggerKernel Graphical Debugger*Appendix I Exported SymbolsKobjects, Ksets, and SubsystemsDMA I/OEasyLDD is best book for beginners to start learning Device Driver programming from basics. Anyone can just take a book and start programming.Book is easy to understand and friendly to use as book has easy language and screenshot of actual output window along with detailed explanation of each program.This book is integration of Author's experimental programs, Latest programming concepts like USB3.0,Contains reference points from all Linux device Driver books and magazines.Book has also collection of many programs available over websites, books and Linux community programs.This book is first milestone towards learning driver programming in step-wise approach.Book will build confidence in you so that you can easily jump in to any type of driver and start coding.All the Best !

Describes the concepts of programming with Linux, covering such topics as shell programming, file structure, managing memory, using MySQL, debugging, processes and signals, and GNOME.

This book provides an overview of modern boot firmware, including the Unified Extensible Firmware Interface (UEFI) and its associated EFI Developer Kit II (EDKII) firmware. The authors have each made significant contributions to developments in these areas. The reader will learn to use the latest developments in UEFI on modern hardware, including open source firmware and open hardware designs. The book begins with an exploration of interfaces exposed to higher-level software and operating systems, and commences to the left of the boot timeline, describing the flow of typical systems, beginning with the machine restart event. Software engineers working with UEFI will benefit greatly from this book, while specific sections of the book address topics relevant for a general audience: system architects, pre-operating-system application developers, operating system vendors (loader, kernel), independent hardware vendors (such as for plug-in

adapters), and developers of end-user applications. As a secondary audience, project technical leaders or managers may be interested in this book to get a feel for what their engineers are doing. The reader will find: An overview of UEFI and underlying Platform Initialization (PI) specifications How to create UEFI applications and drivers Workflow to design the firmware solution for a modern platform Advanced usages of UEFI firmware for security and manageability

Create user-kernel interfaces, work with peripheral I/O, and handle hardware interrupts

Mastering Linux Device Driver Development

Linux Kernel and Device Driver Programming

Linux All-in-One Desk Reference For Dummies

The Definitive Guide to the Xen Hypervisor

Everything you need to start with device driver development for Linux kernel and embedded

Linux

*This handbook gives comprehensive coverage of all kinds of industrial control systems to help engineers and researchers correctly and efficiently implement their projects. It is an indispensable guide and references for anyone involved in control, automation, computer networks and robotics in industry and academia alike. Whether you are part of the manufacturing sector, large-scale infrastructure systems, or processing technologies, this book is the key to learning and implementing real time and distributed control applications. It covers working at the device and machine level as well as the wider environments of plant and enterprise. It includes information on sensors and actuators; computer hardware; system interfaces; digital controllers that perform programs and protocols; the embedded applications software; data communications in distributed control systems; and the system routines that make control systems more user-friendly and safe to operate. This handbook is a single source reference in an industry with highly disparate information from myriad sources. * Helps engineers and researchers correctly and efficiently implement their projects. * An indispensable guide and references for anyone involved in control, automation, computer networks and robotics. * Equally suitable for industry and academia Linux® is being adopted by an increasing number of embedded systems developers, who have been won over by its sophisticated scheduling and networking, its cost-free license, its open development model, and the support offered by rich and powerful programming tools. While there is a great deal of hype surrounding the use of Linux in embedded systems, there is not a lot of practical information. Building Embedded Linux Systems is the first in-depth, hard-core guide to putting together an embedded system based on the Linux kernel. This indispensable*

book features arcane and previously undocumented procedures for: Building your own GNU development toolchain Using an efficient embedded development framework Selecting, configuring, building, and installing a target-specific kernel Creating a complete target root filesystem Setting up, manipulating, and using solid-state storage devices Installing and configuring a bootloader for the target Cross-compiling a slew of utilities and packages Debugging your embedded system using a plethora of tools and techniques Details are provided for various target architectures and hardware configurations, including a thorough review of Linux's support for embedded hardware. All explanations rely on the use of open source and free software packages. By presenting how to build the operating system components from pristine sources and how to find more documentation or help, this book greatly simplifies the task of keeping complete control over one's embedded operating system, whether it be for technical or sound financial reasons. Author Karim Yaghmour, a well-known designer and speaker who is responsible for the Linux Trace Toolkit, starts by discussing the strengths and weaknesses of Linux as an embedded operating system. Licensing issues are included, followed by a discussion of the basics of building embedded Linux systems. The configuration, setup, and use of over forty different open source and free software packages commonly used in embedded Linux systems are also covered. uClibc, BusyBox, U-Boot, OpenSSH, tftpd, tftp, strace, and gdb are among the packages discussed.

Device drivers make it possible for your software to communicate with your hardware, and because every operating system has specific requirements, driver writing is nontrivial. When developing for FreeBSD, you've probably had to scour the Internet and dig through the kernel sources to figure out how to write the drivers you need. Thankfully, that stops now. In FreeBSD Device Drivers, Joseph Kong will teach you how to master everything from the basics of building and running loadable kernel modules to more complicated topics like thread synchronization. After a crash course in the different FreeBSD driver frameworks, extensive tutorial sections dissect real-world drivers like the parallel port printer driver. You'll learn: -All about Newbus, the infrastructure used by FreeBSD to manage the hardware devices on your system -How to work with ISA, PCI, USB, and other buses -The best ways to control and communicate with the hardware devices from user space -How to use Direct Memory Access (DMA) for maximum system performance -The inner workings of the virtual null modem terminal driver, the USB printer driver, the Intel PCI Gigabit Ethernet adapter driver, and other important drivers -How to use Common Access Method (CAM) to manage host bus adapters (HBAs) Concise descriptions and extensive annotations walk you through the many code examples. Don't waste time searching man pages or digging through the kernel sources to figure out how to make that arcane bit of hardware work with your system. FreeBSD Device Drivers gives you the framework that you need to write any driver you want, now.

PLEASE PROVIDE DESCRIPTION

From I/O Ports to Process Management

PCI System Architecture

A Guide for the Intrepid

Development Best Practices for the Internet of Things

Embedded Linux Primer

A Handbook for Engineers and Researchers

Up-to-the-Minute, Complete Guidance for Developing Embedded Solutions with Linux Linux has emerged as today's #1 operating system for embedded products. Christopher Hallinan's Embedded Linux Primer has proven itself as the definitive real-world guide to building efficient, high-value, embedded systems with Linux. Now, Hallinan has thoroughly updated this highly praised book for the newest Linux kernels, capabilities, tools, and hardware support, including advanced multicore processors. Drawing on more than a decade of embedded Linux experience, Hallinan helps you rapidly climb the learning curve, whether you're moving from legacy environments or you're new to embedded programming. Hallinan addresses today's most important development challenges and demonstrates how to solve the problems you're most likely to encounter. You'll learn how to build a modern, efficient embedded Linux development environment, and then utilize it as productively as possible. Hallinan offers up-to-date guidance on everything from kernel configuration and initialization to bootloaders, device drivers to file systems, and BusyBox utilities to real-time configuration and system analysis. This edition adds entirely new chapters on UDEV, USB, and open source build systems. Tour the typical embedded system and development environment and understand its concepts and components. Understand the Linux kernel and userspace initialization processes. Preview bootloaders, with specific emphasis on U-Boot. Configure the Memory Technology Devices (MTD) subsystem to interface with flash (and other) memory devices. Make the most of BusyBox and latest open source development tools. Learn from expanded and updated coverage of kernel debugging. Build and analyze real-time systems with Linux. Learn to configure device files and driver loading with UDEV. Walk through detailed coverage of the USB subsystem. Introduces the latest open source embedded Linux build systems. Reference appendices include U-Boot and

BusyBox commands.

*Linux Kernel Module Programming Guide is for people who want to write kernel modules. It takes a hands-on approach starting with writing a small "hello, world" program, and quickly moves from there. Far from a boring text on programming, Linux Kernel Module Programming Guide has a lively style that entertains while it educates. An excellent guide for anyone wishing to get started on kernel module programming. *** Money raised from the sale of this book supports the development of free software and documentation. Presents an overview of kernel configuration and building for version 2.6 of the Linux kernel.*

Over the last few years, Linux has grown both as an operating system and a tool for personal and business use. Simultaneously becoming more user friendly and more powerful as a back-end system, Linux has achieved new plateaus: the newer filesystems have solidified, new commands and tools have appeared and become standard, and the desktop--including new desktop environments--have proved to be viable, stable, and readily accessible to even those who don't consider themselves computer gurus. Whether you're using Linux for personal software projects, for a small office or home office (often termed the SOHO environment), to provide services to a small group of colleagues, or to administer a site responsible for millions of email and web connections each day, you need quick access to information on a wide range of tools. This book covers all aspects of administering and making effective use of Linux systems. Among its topics are booting, package management, and revision control. But foremost in Linux in a Nutshell are the utilities and commands that make Linux one of the most powerful and flexible systems available. Now in its fifth edition, Linux in a Nutshell brings users up-to-date with the current state of Linux. Considered by many to be the most complete and authoritative command reference for Linux available, the book covers all substantial user, programming, administration, and networking commands for the most common Linux distributions. Comprehensive but concise, the fifth edition has been updated to cover new features of major Linux distributions. Configuration information for the rapidly growing commercial network services and community update services is one of the subjects covered

for the first time. But that's just the beginning. The book covers editors, shells, and LILO and GRUB boot options. There's also coverage of Apache, Samba, Postfix, sendmail, CVS, Subversion, Emacs, vi, sed, gawk, and much more. Everything that system administrators, developers, and power users need to know about Linux is referenced here, and they will turn to this book again and again.

Where the Kernel Meets the Hardware

Embedded Firmware Solutions

Linux Kernel in a Nutshell

Security on z/VM

Linux Device Driver Development

Easy Linux Device Driver, Second Edition

To thoroughly understand what makes Linux tick and why it's so efficient, you need to delve deep into the heart of the operating system--into the Linux kernel itself. The kernel is Linux--in the case of the Linux operating system, it's the only bit of software to which the term "Linux" applies. The kernel handles all the requests or completed I/O operations and determines which programs will share its processing time, and in what order. Responsible for the sophisticated memory management of the whole system, the Linux kernel is the force behind the legendary Linux efficiency. The new edition of **Understanding the Linux Kernel** takes you on a guided tour through the most significant data structures, many algorithms, and programming tricks used in the kernel. Probing beyond the superficial features, the authors offer valuable insights to people who want to know how things really work inside their machine. Relevant segments of code are dissected and discussed line by line. The book covers more than just the functioning of the code, it explains the theoretical underpinnings for why Linux does things the way it does. The new edition of the book has been updated to cover version 2.4 of the kernel, which is quite different from version 2.2: the virtual memory system is entirely new, support for multiprocessor systems is improved, and whole new classes of hardware devices have been added. The authors explore each new feature in detail. Other topics in the book include: Memory management including file buffering, process swapping, and Direct memory Access (DMA) The Virtual Filesystem and the Second Extended Filesystem Process creation and scheduling Signals, interrupts, and the essential interfaces to device drivers Timing Synchronization in the kernel Interprocess Communication (IPC) Program execution **Understanding the Linux Kernel, Second Edition** will acquaint you with all the inner workings of Linux, but is more than just an academic exercise. You'll learn what conditions bring out Linux's best performance, and you'll see how it meets the challenge of providing good system response during process scheduling, file access, and memory management in a wide

variety of environments. If knowledge is power, then this book will help you make the most of your Linux system.

Provides information on using the Linux operating system, covering such topics as the desktop, networking, Internet servers, administration, security, and programming.

Nwely updated to include new calls and techniques introduced in Versions 2.2 and 2.4 of the Linux kernel, a definitive resource for those who want to support computer peripherals under the Linux operating system explains how to write a driver for a broad spectrum of devices, including character devices, network interfaces, and block devices. Original.

(Intermediate)

Embedded Firmware Solutions is the perfect introduction and daily-use field guide--for the thousands of firmware designers, hardware engineers, architects, managers, and developers--to Intel's new firmware direction (including Quark coverage), showing how to integrate Intel® Architecture designs into their plans. Featuring hands-on examples and exercises using Open Source codebases, like Coreboot and EFI Development Kit (tianocore) and Chromebook, this is the first book that combines a timely and thorough overview of firmware solutions for the rapidly evolving embedded ecosystem with in-depth coverage of requirements and optimization.

Linux All-in-One For Dummies

Data Plane Development Kit (DPDK)

A comprehensive guide to kernel internals, writing kernel modules, and kernel synchronization

Concepts, Techniques, Tricks, and Traps

Algorithms and Structures of Version 2.4

Developing with the Unified Extensible Firmware Interface, Third Edition

Get under the hood of Xen, the high performance virtualization software.

Easy Linux Device Driver : First Step Towards Device Driver Programming Easy Linux Device Driver book is an easy and friendly way of learning device driver programming . Book contains all latest programs along with output screen screenshots. Highlighting important sections and stepwise approach helps for quick understanding of programming . Book contains Linux installation ,Hello world program up to USB 3.0 ,Display Driver ,PCI device driver programming concepts in stepwise approach. Program gives best understanding of theoretical and practical fundamentals of Linux device driver. Beginners should start learning Linux device driver from this book to become device driver expertise. Topics covered: Introduction of Linux Advantages of Linux History of Linux Architecture of Linux Definations Ubuntu installation Ubuntu Installation Steps User Interface Difference About KNOPPIX Important links Terminal: Soul of Linux Creating Root account Terminal Commands Virtual Editor Commands Linux Kernel Linux Kernel Internals Kernel Space and User space Device

Driver Place of Driver in System Device Driver working Characteristics of Device Driver Module Commands Hello World Program pre-settings Write Program Printk function Makefile Run program Parameter passing Parameter passing program Parameter Array Process related program Process related program Character Device Driver Major and Minor number API to registers a device Program to show device number Character Driver File Operations File operation program. Include .h header Functions in module.h file Important code snippets Summary of file operations PCI Device Driver Direct Memory Access Module Device Table Code for Basic Device Driver Important code snippets USB Device Driver Fundamentals Architecture of USB device driver USB Device Driver program Structure of USB Device Driver Parts of USB end points Important features USB information Driver USB device Driver File Operations Using URB Simple data transfer Program to read and write Important code snippets Gadget Driver Complete USB Device Driver Program Skeleton Driver Program Special USB 3.0 USB 3.0 Port connection Bulk endpoint streaming Stream ID Device Driver Lock Mutual Exclusion Semaphore Spin Lock Display Device Driver Frame buffer concept Framebuffer Data Structure Check and set Parameter Accelerated Method Display Driver summary Memory Allocation Kmalloc Vmalloc Ioremap Interrupt Handling interrupt registration Proc interface Path of interrupt Programming Tips Softirqs, Tasklets, Work Queues I/O Control Introducing ioctl Prototype Stepwise execution of ioctl Sample Device Driver Complete memory Driver Complete Parallel Port Driver Device Driver Debugging Data Display Debugger Graphical Display Debugger Kernel Graphical Debugger Appendix I Exported Symbols Kobjects, Ksets, and Subsystems DMA I/O

This book is written for students or professionals who quickly want to learn Linux Kernel programming and device driver development. Each chapter in this book is associated with code samples and code commentary so that the readers may quickly un.

Learn to develop customized device drivers for your embedded Linux system About This Book Learn to develop customized Linux device drivers Learn the core concepts of device drivers such as memory management, kernel caching, advanced IRQ management, and so on. Practical experience on the embedded side of Linux Who This Book Is For This book will help anyone who wants to get started with developing their own Linux device drivers for embedded systems. Embedded Linux users will benefit highly from this book. This book covers all about device driver development, from char drivers to network device drivers to memory management. What You Will Learn Use kernel facilities to develop powerful drivers Develop drivers for widely used I2C and SPI devices and use the regmap API Write and support devicetree from within your drivers Program advanced drivers for network and frame buffer devices Delve into the Linux irqdomain API and write interrupt controller drivers Enhance your skills with regulator and PWM frameworks Develop measurement system drivers with IIO framework Get the best from memory management and the DMA subsystem Access and manage GPIO

subsystems and develop GPIO controller drivers In Detail Linux kernel is a complex, portable, modular and widely used piece of software, running on around 80% of servers and embedded systems in more than half of devices throughout the World. Device drivers play a critical role in how well a Linux system performs. As Linux has turned out to be one of the most popular operating systems used, the interest in developing proprietary device drivers is also increasing steadily. This book will initially help you understand the basics of drivers as well as prepare for the long journey through the Linux Kernel. This book then covers drivers development based on various Linux subsystems such as memory management, PWM, RTC, IIO, IRQ management, and so on. The book also offers a practical approach on direct memory access and network device drivers. By the end of this book, you will be comfortable with the concept of device driver development and will be in a position to write any device driver from scratch using the latest kernel version (v4.13 at the time of writing this book). Style and approach A set of engaging examples to develop Linux device drivers

Linux in a Nutshell

A Practical Real-World Approach

A Software Optimization Guide to the User Space-Based Network Applications

Algorithms and Architectures for Parallel Processing

Linux Device Drivers

Linux Kernel Development

This book brings together the insights and practical experience of some of the most experienced Data Plane Development Kit (DPDK) technical experts, detailing the trend of DPDK, data packet processing, hardware acceleration, packet processing and virtualization, as well as the practical application of DPDK in the fields of SDN, NFV, and network storage. The book also devotes many chunks to exploring various core software algorithms, the advanced optimization methods adopted in DPDK, detailed practical experience, and the guides on how to use DPDK.

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 learn

Get to grips with the basics of the modern Linux Device Model (LDM) Write a simple yet complete misc class character device driver Perform user-kernel interfacing using popular methods Understand and handle hardware interrupts confidently Perform I/O on peripheral hardware chip memory Explore kernel APIs to work with delays, timers, kthreads, and workqueues Understand kernel concurrency issues Work with key kernel synchronization primitives and discover how to detect and avoid deadlock Who 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.

In order to thoroughly understand what makes Linux tick and why it works so well on a wide variety of systems, you need to delve deep into the heart of the kernel. The kernel handles all interactions between the CPU and the external world, and determines which programs will share processor time, in what order. It manages limited memory so well that hundreds of processes can share the system efficiently, and expertly organizes data transfers so that the CPU isn't kept waiting any longer than necessary for the relatively slow disks. The third edition of Understanding the Linux Kernel takes you on a guided tour of the most significant data structures, algorithms, and programming tricks used in the kernel. Probing beyond superficial features, the authors offer valuable insights to people who want to know how things really work inside their machine. Important Intel-specific features are discussed. Relevant segments of code are dissected line by line. But the book covers more than just the functioning of the code; it explains the theoretical underpinnings of why Linux does things the way it does. This edition of the book covers Version 2.6, which has seen significant changes to nearly every kernel subsystem, particularly in the areas of memory management and block devices. The book focuses on the following topics: Memory management, including file buffering, process swapping, and Direct memory Access (DMA) The Virtual Filesystem layer and the Second and Third Extended Filesystems Process creation and scheduling Signals, interrupts, and the essential interfaces to device drivers Timing Synchronization within the kernel Interprocess Communication (IPC) Program execution Understanding the Linux Kernel will acquaint you with all the inner workings of Linux, but it's more than just an academic exercise. You'll learn what conditions bring out Linux's best performance, and you'll see how it meets the challenge of providing good system response during process scheduling, file access, and memory management in a wide variety of environments. This book will help you make the most of your Linux system.

Master the art of developing customized device drivers for your embedded Linux systems Key Features Stay up to date with the Linux PCI, ASoC, and V4L2 subsystems and write device drivers for them Get to grips with the Linux kernel power management infrastructure Adopt a practical approach to customizing your Linux environment using best practices Book Description Linux is one of

the fastest-growing operating systems around the world, and in the last few years, the Linux kernel has evolved significantly to support a wide variety of embedded devices with its improved subsystems and a range of new features. With this book, you'll find out how you can enhance your skills to write custom device drivers for your Linux operating system. Mastering Linux Device Driver Development provides complete coverage of kernel topics, including video and audio frameworks, that usually go unaddressed. You'll work with some of the most complex and impactful Linux kernel frameworks, such as PCI, ALSA for SoC, and Video4Linux2, and discover expert tips and best practices along the way. In addition to this, you'll understand how to make the most of frameworks such as NVMEM and Watchdog. Once you've got to grips with Linux kernel helpers, you'll advance to working with special device types such as Multi-Function Devices (MFD) followed by video and audio device drivers. By the end of this book, you'll be able to write feature-rich device drivers and integrate them with some of the most complex Linux kernel frameworks, including V4L2 and ALSA for SoC. What you will learnExplore and adopt Linux kernel helpers for locking, work deferral, and interrupt managementUnderstand the Regmap subsystem to manage memory accesses and work with the IRQ subsystemGet to grips with the PCI subsystem and write reliable drivers for PCI devicesWrite full multimedia device drivers using ALSA SoC and the V4L2 frameworkBuild power-aware device drivers using the kernel power management frameworkFind out how to get the most out of miscellaneous kernel subsystems such as NVMEM and WatchdogWho this book is for This book is for embedded developers, Linux system engineers, and system programmers who want to explore Linux kernel frameworks and subsystems. C programming skills and a basic understanding of driver development are necessary to get started with this book.

The Linux Kernel Module Programming Guide

Beginning Linux?Programming

16th International Conference, ICA3PP 2016, Granada, Spain, December 14-16, 2016, Proceedings

Write custom device drivers to support computer peripherals in Linux operating systems

Provides a definitive resource for those who want to support computer peripherals under the Linux operating system, explaining how to write a driver for a broad spectrum of devices, including character devices, network interfaces, and block devices. Original. (Intermediate). This book constitutes the refereed proceedings of the 16th International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP 2016, held in Granada, Spain, in December 2016. The 30 full papers and 22 short papers presented were carefully reviewed and selected from 117 submissions. They cover many dimensions of parallel algorithms and architectures, encompassing fundamental theoretical approaches, practical experimental projects, and commercial components and systems trying to push beyond the limits of existing technologies, including experimental efforts, innovative systems, and investigations that identify weaknesses in existing parallel processing technology.

There's a great deal of excitement surrounding the use of Linux in embedded systems -- for

everything from cell phones to car ABS systems and water-filtration plants -- but not a lot of practical information. Building Embedded Linux Systems offers an in-depth, hard-core guide to putting together embedded systems based on Linux. Updated for the latest version of the Linux kernel, this new edition gives you the basics of building embedded Linux systems, along with the configuration, setup, and use of more than 40 different open source and free software packages in common use. The book also looks at the strengths and weaknesses of using Linux in an embedded system, plus a discussion of licensing issues, and an introduction to real-time, with a discussion of real-time options for Linux. This indispensable book features arcane and previously undocumented procedures for:

- Building your own GNU development toolchain
- Using an efficient embedded development framework
- Selecting, configuring, building, and installing a target-specific kernel
- Creating a complete target root filesystem
- Setting up, manipulating, and using solid-state storage devices
- Installing and configuring a bootloader for the target
- Cross-compiling a slew of utilities and packages
- Debugging your embedded system using a plethora of tools and techniques
- Using the uClibc, BusyBox, U-Boot, OpenSSH, tftpd, tftp, strace, and gdb packages

By presenting how to build the operating system components from pristine sources and how to find more documentation or help, Building Embedded Linux Systems greatly simplifies the task of keeping complete control over your embedded operating system.