

Download Free Linux Kernel Development 2nd Edition Read Pdf Free

Linux Kernel Development [Mastering Linux Kernel Development](#) [Linux Kernel Development](#) [Linux Kernel Programming](#) [Linux System Programming](#) [Mastering Linux Kernel Development](#) [Professional Linux Kernel Architecture](#) [Linux Kernel Programming Part 2 - Char Device Drivers and Kernel Synchronization](#) [Linux Device Drivers](#) [Windows Kernel Programming](#) [A Guide to Kernel Exploitation](#) [Lateral Kernel Development in the Progeny of Six Rowed - Two Rowed Barley Crosses](#) [Understanding the Linux Kernel](#) [Linux Kernel in a Nutshell](#) [Linux Device Drivers](#) [Mastering Embedded Linux Programming](#) [Proceedings of the 2nd Annual International Conference on Material, Machines and Methods for Sustainable Development \(MMMS2020\)](#) [The Linux Kernel Module Programming Guide](#) [Linux Administration Handbook](#) [Beginning Linux? Programming](#) [Understanding the Linux Kernel](#) [Linux System Programming](#) [Linux Kernel Internals](#) [Linux in a Nutshell](#) [Maize Kernel Development](#) [Advanced Operating Systems and Kernel Applications: Techniques and Technologies](#) [The Linux Kernel Primer](#) [The Physiology of Vegetable Crops, 2nd Edition](#) [Designing BSD Rootkits](#) [Pro Git](#) [Seed Biology and Yield of Grain Crops, 2nd Edition](#) [Advances in Computer Systems Architecture](#) [Programming Languages and Systems](#) [The Art of Assembly Language, 2nd Edition](#) [How Linux Works, 2nd Edition](#) [Embedded Android](#) [How Linux Works](#) [MicroC/OS-II Encyclopedia of Computer Science and Technology](#) [Parallel Processing and Applied Mathematics, Part 1](#)

This practical guide helps programmers better understand the Linux kernel, and to write and develop kernel code. It provides in-depth coverage of all the major subsystems and features of the Linux 2.6 kernel. "This book discusses non-distributed operating systems that benefit researchers, academicians, and practitioners"--Provided by publisher. 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. 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. 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 "hands-on" information on writing device drivers for the Linux system, with particular focus on the features of the 2.4 kernel and its implementation Embedded Android is for Developers wanting to create embedded systems based on Android and for those wanting to port Android to new hardware, or creating a custom development environment. Hackers and moders will also find this an indispensable guide to how Android works. UNIX, UNIX LINUX & UNIX TCL/TK. Write software that makes the most effective use of the Linux system, including the kernel and core system libraries. The majority of both Unix and Linux code is still written at the system level, and this book helps you focus on everything above the kernel, where applications such as Apache, bash, cp, vim, Emacs, gcc, gdb, glibc, ls, mv, and X exist. Written primarily for engineers looking to program at the low level, this updated edition of Linux System Programming gives you an understanding of core internals that makes for better code, no matter where it appears in the stack. -- Provided by publisher. This book is about writing software that makes the most effective use of the system you're running on -- code that interfaces directly with the kernel and core system libraries, including the shell, text editor, compiler, debugger, core utilities, and system daemons. The majority of both Unix and Linux code is still written at the system level, and Linux System Programming focuses on everything above the kernel, where applications such as Apache, bash, cp, vim, Emacs, gcc, gdb, glibc, ls, mv, and X exist. Written primarily for engineers looking to program (better) at the low level, this book is an ideal teaching tool for any programmer. Even with the trend toward high-level development, either through web software (such as PHP) or managed code (C#), someone still has to write the PHP interpreter and the C# virtual machine. Linux System Programming gives you an understanding of core internals that makes for better code, no matter where it appears in the stack. Debugging high-level code often requires you to understand the system calls and kernel behavior of your operating system, too. Key topics include: An overview of Linux, the kernel, the C library, and the C compiler Reading from and writing to files, along with other basic file I/O operations, including how the Linux kernel implements and manages file I/O Buffer size management, including the Standard I/O library 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 File and directories-creating, moving, copying, deleting, and managing them Memory management -- interfaces for allocating memory, managing the memory youhave, and optimizing your memory access Signals and their role on a Unix system, plus basic and advanced signal interfaces Time, sleeping, and clock management, starting with the basics and continuing through POSIX clocks and high resolution timers With Linux System Programming, you will be able to take an in-depth look at Linux from both a theoretical and an applied perspective as you cover a wide range of programming topics. There is nothing like the power of the kernel in Windows - but how do you write kernel drivers to take advantage of that power? This book will show you how.The book describes software kernel drivers programming for Windows. These drivers don't deal with hardware, but rather with the system itself: processes, threads, modules, registry and more. Kernel code can be used for monitoring important events, preventing some from occurring if needed. Various filters can be written that can intercept calls that a driver may be interested in. Presents an illustrated A-Z encyclopedia containing approximately 600 entries on computer and technology related topics. The refereed proceedings of the 12th Asia-Pacific Computer Systems Architecture Conference are presented in this volume. Twenty-six full papers are presented together with two keynote and eight invited lectures. Collectively, they represent some of the most important developments in computer systems architecture. The papers emphasize hardware and software techniques for state-of-the-art, multi-core and multi-threaded architectures. 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. Assembly is a low-level programming language that's one step above a computer's native machine language. Although assembly language is commonly used for writing device drivers, emulators, and video games, many programmers find its somewhat unfriendly syntax intimidating to learn and use. Since 1996, Randall Hyde's *The Art of Assembly Language* has provided a comprehensive, plain-English, and patient introduction to 32-bit x86 assembly for non-assembly programmers. Hyde's primary teaching tool, High Level Assembler (or HLA), incorporates many of the features found in high-level languages (like C, C++, and Java) to help you quickly grasp basic assembly concepts. HLA lets you write true low-level code while enjoying the benefits of high-level language programming. As you read *The Art of Assembly Language*, you'll learn the low-level theory fundamental to computer science and turn that understanding into real, functional code. You'll learn how to:

- Edit, compile, and run HLA programs
- Declare and use constants, scalar variables, pointers, arrays, structures, unions, and namespaces
- Translate arithmetic expressions (integer and floating point)
- Convert high-level control structures

This much anticipated second edition of *The Art of Assembly Language* has been updated to reflect recent changes to HLA and to support Linux, Mac OS X, and FreeBSD. Whether you're new to programming or you have experience with high-level languages, *The Art of Assembly Language, 2nd Edition* is your essential guide to learning this complex, low-level language. This new edition of an established title examines the determination of grain crop yield from a unique perspective, by concentrating on the influence of the seed itself. As the food supply for an expanding world population is based on grain crops harvested for their seeds, understanding the process of seed growth and its regulation is crucial to our efforts to increase production and meet the needs of that population. Yield of grain crops is determined by their assimilatory processes such as photosynthesis and the biosynthetic processes in the seed, which are partly regulated within the seed itself. Substantially updated with new research and further developments of the practical applications of the concepts explored, this book is essential reading for those concerned with seed science and crop yield, including agronomists, crop physiologists, plant breeders, and extension workers. It is also a valuable source of information for lecturers and graduate students of agronomy and plant physiology.

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. Contains an introduction to the operating system with detailed documentation on commands, utilities, programs, system configuration, and networking. This is an authoritative book that acts as a guide to understanding maize kernel development. Written by a team of experts, it covers topics spanning pre- and post-fertilization events, embryo and endosperm development, grain filling and maturation, and factors influencing crop yield. It explores the significance of maize and other cereal grains, existing hypotheses and research, and important gaps in our knowledge and how we might fill them. This is a valuable resource for researchers of maize and other cereals, and anyone working on basic or applied science in the fields of seed development, plant genetics, and crop physiology. Provides information on writing a driver in Linux, covering such topics as character devices, network interfaces, driver debugging, concurrency, and interrupts. "Designing BSD Rootkits" introduces the fundamentals of programming and developing rootkits under the FreeBSD operating system. Written in a friendly, accessible style and sprinkled with geek humor and pop culture references, the author favors a "learn by example" approach that assumes no prior kernel hacking experience. Git is the version control system developed by Linus Torvalds for Linux kernel development. It took the open source world by storm since its inception in 2005, and is used by small development shops and giants like Google, Red Hat, and IBM, and of course many open source projects. A book by Git experts to turn you into a Git expert Introduces the world of distributed version control Shows how to build a Git development workflow This book constitutes the refereed proceedings of the 10th Asian Symposium on Programming Languages and Systems, APLAS 2012, held in Kyoto, Japan, in December 2012. The 24 revised full papers presented together with the abstracts of 3 invited talks were carefully reviewed and selected from 58 submissions. The papers are organized in topical sections on concurrency, security, static analysis, language design, dynamic analysis, complexity and semantics, and program logics and verification. A Guide to Kernel Exploitation: Attacking the Core discusses the theoretical techniques and approaches needed to develop reliable and effective kernel-level exploits, and applies them to different operating systems, namely, UNIX derivatives, Mac OS X, and Windows. Concepts and tactics are presented categorically so that even when a specifically detailed vulnerability has been patched, the foundational information provided will help hackers in writing a newer, better attack; or help pen testers, auditors, and the like develop a more concrete design and defensive structure. The book is organized into four parts. Part I introduces the kernel and sets out the theoretical basis on which to build the rest of the book. Part II focuses on different operating systems and describes exploits for them that target various bug classes. Part III on remote kernel exploitation analyzes the effects of the remote scenario and presents new techniques to target remote issues. It includes a step-by-step analysis of the development of a reliable, one-shot, remote exploit for a real vulnerability bug affecting the SCTP subsystem found in the Linux kernel. Finally, Part IV wraps up the analysis on kernel exploitation and looks at what the future may hold. Covers a range of operating system families — UNIX derivatives, Mac OS X, Windows Details common scenarios such as generic memory corruption (stack overflow, heap overflow, etc.) issues, logical bugs and race conditions Delivers the reader from user-land exploitation to the world of kernel-land (OS) exploits/attacks, with a particular focus on the steps that lead to the creation of successful techniques, in order to give to the reader something more than just a set of tricks MicroC/OS II Second Edition describes the design and implementation of the MicroC/OS-II real-time operating system (RTOS). In addition to its value as a reference to the kernel, it is an extremely detailed and highly readable design study particularly useful to the embedded systems student. While documenting the design and implementation of the kernel Offers a comprehensive view of the underpinnings of the Linux kernel on the Intel x86 and the Power PC. How Linux Works describes the inside of the Linux system for systems administrators, whether they maintain an extensive network in the office or one Linux box at home. After a guided tour of filesystems, the boot sequence, system management basics, and networking, author Brian Ward delves into topics such as development tools, custom kernels, and buying hardware. With a mixture of background theory and real-world examples, this book shows both how to administer Linux, and why each particular technique works, so that you will know how to make Linux work for you. Find an introduction to the architecture, concepts and algorithms of the Linux kernel in Professional Linux Kernel Architecture, a guide to the kernel sources and large number of connections among subsystems. Find an introduction to the relevant structures and functions exported by the kernel to userland, understand the theoretical and conceptual aspects of the Linux kernel and Unix derivatives, and gain a deeper understanding of the kernel. Learn how to reduce the vast amount of information contained in the kernel sources and obtain the skills necessary to understand the kernel sources. Unlike some operating systems, Linux doesn't try to hide the important bits from you—it gives you full control of your computer. But to truly master Linux, you need to understand its internals, like how the system boots, how networking works, and what the kernel actually does. In this completely revised second edition of the perennial best seller How Linux Works, author Brian Ward makes the concepts behind Linux internals accessible to anyone curious about the inner workings of the operating system. Inside, you'll find the kind of knowledge that normally comes from years of experience doing things the hard way. You'll learn: –How Linux boots, from boot

loaders to init implementations (systemd, Upstart, and System V) –How the kernel manages devices, device drivers, and processes –How networking, interfaces, firewalls, and servers work –How development tools work and relate to shared libraries –How to write effective shell scripts You'll also explore the kernel and examine key system tasks inside user space, including system calls, input and output, and filesystems. With its combination of background, theory, real-world examples, and patient explanations, How Linux Works will teach you what you need to know to solve pesky problems and take control of your operating system. This book constitutes the proceedings of the 8th International Conference on Parallel Processing and Applied Mathematics, PPAM 2009, held in Wroclaw, Poland, in September 2009. Completely updated and revised, this bestselling book continues to explain the growth and developmental processes involved in the formation of vegetables. Since the publication of the successful first edition significant discoveries, particularly in the area of molecular biology, have deepened and broadened our knowledge and understanding of these processes. This new edition brings the topic up-to-date and is presented over two sections: the first provides general knowledge on germination, transplanting, flowering, the effects of stress and modelling, whilst the second section details the physiology of specific crops or crop groups. 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. Presents an overview of kernel configuration and building for version 2.6 of the Linux kernel. "As this book shows, Linux systems are just as functional, secure, and reliable as their proprietary counterparts. Thanks to the ongoing efforts of thousands of Linux developers, Linux is more ready than ever for deployment at the frontlines of the real world. The authors of this book know that terrain well, and I am happy to leave you in their most capable hands." –Linus Torvalds "The most successful sysadmin book of all time—because it works!" –Rik Farrow, editor of :login: "This book clearly explains current technology with the perspective of decades of experience in large-scale system administration. Unique and highly recommended." –Jonathan Corbet, cofounder, LWN.net "Nemeth et al. is the overall winner for Linux administration: it's intelligent, full of insights, and looks at the implementation of concepts." –Peter Salus, editorial director, Matrix.net Since 2001, Linux Administration Handbook has been the definitive resource for every Linux® system administrator who must efficiently solve technical problems and maximize the reliability and performance of a production environment. Now, the authors have systematically updated this classic guide to address today's most important Linux distributions and most powerful new administrative tools. The authors spell out detailed best practices for every facet of system administration, including storage management, network design and administration, web hosting, software configuration management, performance analysis, Windows interoperability, and much more. Sysadmins will especially appreciate the thorough and up-to-date discussions of such difficult topics such as DNS, LDAP, security, and the management of IT service organizations. Linux® Administration Handbook, Second Edition, reflects the current versions of these leading

distributions: Red Hat® Enterprise Linux® Fedora™ Core SUSE® Linux Enterprise Debian® GNU/Linux Ubuntu® Linux Sharing their war stories and hard-won insights, the authors capture the behavior of Linux systems in the real world, not just in ideal environments. They explain complex tasks in detail and illustrate these tasks with examples drawn from their extensive hands-on experience. 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 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 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. The book starts with the basics, explaining how to compile and run your first program. First, each concept is explained to give you a solid understanding of the material. Practical examples are then presented, so you see how to apply the knowledge in real applications. Aimed at programmers, this book shows how the Linux operating system actually works so that they can start to program the Linux kernel. The CD-ROM contains the Slackware distribution 3.1 together with its complete source code and much more. Harness the power of Linux to create versatile and robust embedded solutions

Key Features Learn how to develop and configure robust embedded Linux devices Explore the new features of Linux 5.4 and the Yocto Project 3.1 (Dunfell) Discover different ways to debug and profile your code in both user space and the Linux kernel

Book Description If you're looking for a book that will demystify embedded Linux, then you've come to the right place. Mastering Embedded Linux Programming is a fully comprehensive guide that can serve both as a means to learn new things or as a handy reference. The first few chapters of this book will break down the fundamental elements that underpin all embedded Linux projects: the toolchain, the bootloader, the kernel, and the root filesystem. After that, you will learn how to create each of these elements from scratch and automate the process using Buildroot and the Yocto Project. As you progress, the book will show you how to implement an effective storage strategy for flash memory chips and install updates to a device remotely once it's deployed. You'll also learn about the key aspects of writing code for embedded Linux, such as how to access hardware from apps, the implications of writing multi-threaded code, and techniques to manage memory in an efficient way. The final chapters demonstrate how to debug your code, whether it resides in apps or in the Linux kernel itself. You'll also cover the different tracers and profilers that are available for Linux so that you can quickly pinpoint any performance bottlenecks in your system. By the end of this Linux book, you'll be able to create efficient and secure embedded devices using Linux. What you will learn

Use Buildroot and the Yocto Project to create embedded Linux systems Troubleshoot BitBake build failures and streamline your Yocto development workflow Update IoT devices securely in the field using Mender or balena Prototype peripheral additions by reading schematics, modifying device trees, soldering breakout boards, and probing pins with a logic analyzer Interact with hardware without having to write kernel device drivers Divide your system up into services supervised by BusyBox

Debug devices remotely using GDB and measure the performance of systems using tools such as perf, ftrace, eBPF, and Callgrind Who this book is for If you're a systems software engineer or system administrator who wants to learn how to implement Linux on embedded devices, then this book is for you. It's also aimed at embedded systems engineers accustomed to programming for low-power microcontrollers, who can use this book to help make the leap to high-speed systems on chips that can run Linux. Anyone who develops hardware that needs to run Linux will find

something useful in this book – but before you get started, you'll need a solid grasp on POSIX standard, C programming, and shell scripting. This book presents selected, peer-reviewed proceedings of the 2nd International Conference on Material, Machines and Methods for Sustainable Development (MMMS2020), held in the city of Nha Trang, Vietnam, from 12 to 15 November, 2020. The purpose of the conference is to explore and ensure an understanding of the critical aspects contributing to sustainable development, especially materials, machines and methods. The contributions published in this book come from authors representing universities, research institutes and industrial companies, and reflect the results of a very broad spectrum of research, from micro- and nanoscale materials design and processing, to mechanical engineering technology in industry. Many of the contributions selected for these proceedings focus on materials modeling, eco-material processes and mechanical manufacturing.

- [Linux Kernel Development](#)
- [Mastering Linux Kernel Development](#)
- [Linux Kernel Development](#)
- [Linux Kernel Programming](#)
- [Linux System Programming](#)
- [Mastering Linux Kernel Development](#)
- [Professional Linux Kernel Architecture](#)
- [Linux Kernel Programming Part 2 Char Device Drivers And Kernel Synchronization](#)
- [Linux Device Drivers](#)
- [Windows Kernel Programming](#)
- [A Guide To Kernel Exploitation](#)
- [Lateral Kernel Development In The Progeny Of Six Rowed Two Rowed Barley Crosses](#)
- [Understanding The Linux Kernel](#)
- [Linux Kernel In A Nutshell](#)
- [Linux Device Drivers](#)
- [Mastering Embedded Linux Programming](#)
- [Proceedings Of The 2nd Annual International Conference On Material Machines And Methods For Sustainable Development MMMS2020](#)
- [The Linux Kernel Module Programming Guide](#)
- [Linux Administration Handbook](#)
- [Beginning Linux Programming](#)
- [Understanding The Linux Kernel](#)
- [Linux System Programming](#)
- [Linux Kernel Internals](#)
- [Linux In A Nutshell](#)
- [Maize Kernel Development](#)
- [Advanced Operating Systems And Kernel Applications Techniques And Technologies](#)
- [The Linux Kernel Primer](#)
- [The Physiology Of Vegetable Crops 2nd Edition](#)
- [Designing BSD Rootkits](#)
- [Pro Git](#)
- [Seed Biology And Yield Of Grain Crops 2nd Edition](#)
- [Advances In Computer Systems Architecture](#)
- [Programming Languages And Systems](#)
- [The Art Of Assembly Language 2nd Edition](#)
- [How Linux Works 2nd Edition](#)
- [Embedded Android](#)
- [How Linux Works](#)
- [MicroC OS II](#)
- [Encyclopedia Of Computer Science And Technology](#)
- [Parallel Processing And Applied Mathematics Part I](#)