

# Professional Linux Programming

Professional Linux Programming John Wiley & Sons

Numerous people still believe that learning and acquiring expertise in Linux is not easy, that only a professional can understand how a Linux system works. Nowadays, Linux has gained much popularity both at home and at the workplace. Linux Yourself: Concept and Programming aims to help and guide people of all ages by offering a deep insight into the concept of Linux, its usage, programming, administration, and several other connected topics in an easy approach. This book can also be used as a textbook for undergraduate/postgraduate engineering students and others who have a passion to gain expertise in the field of computer science/information technology as a Linux developer or administrator. The word "Yourself" in the title refers to the fact that the content of this book is designed to give a good foundation to understand the Linux concept and to guide yourself as a good Linux professional in various platforms. There are no prerequisites to understand the contents from this book, and a person with basic knowledge of C programming language will be able to grasp the concept with ease. With this mindset, all the topics are presented in such a way that it should be simple, clear, and straightforward with many examples and figures. Linux is distinguished by its own power and flexibility, along with open-source accessibility and community as compared to other operating systems, such as Windows and macOS. It is the author's sincere view that readers of all levels will find this book worthwhile and will be able to learn or sharpen their skills. **KEY FEATURES** Provides a deep conceptual learning and expertise in programming skill for any user about Linux, UNIX, and their features. Elaborates GUI and CUI including Linux commands, various shells, and the vi editor Details file management and file systems to understand Linux system architecture easily Promotes hands-on practices of regular expressions and advanced filters, such as sed and awk through many helpful examples Describes an insight view of shell scripting, process, thread, system calls, signal, inter-process communication, X Window System, and many more aspects to understand the system programming in the Linux environment Gives a detailed description of Linux administration by elaborating LILO, GRUB, RPM-based package, and program installation and compilation that can be very helpful in managing the Linux system in a very efficient way Reports some famous Linux distributions to understand the similarity among all popular available Linux and other features as case studies

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. This Linux book begins by showing you how to build the kernel from the source. Next, you'll learn how to write your first kernel module using the powerful Loadable Kernel Module (LKM) framework. The book then covers key kernel internals topics including Linux kernel architecture, memory management, and CPU scheduling. Next, 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. Linux kernel and driver developers looking to overcome frequent and common kernel development issues, as well as understand kernel internals, will benefit from this book. A basic understanding of Linux CLI and C programming is required.

This book is broken into four primary sections addressing key topics that Linux programmers need to master: Linux nuts and bolts, the Linux kernel, the Linux desktop, and Linux for the Web **Effective examples help get readers up to speed with building software on a Linux-based system while using the tools and utilities that contribute to streamlining the software development process** Discusses using emulation and virtualization technologies for kernel development and application testing Includes useful insights aimed at helping readers understand how their applications code fits in with the rest of the software stack Examines cross-compilation, dynamic device insertion and removal, key Linux projects (such as Project Utopia), and the internationalization capabilities present in the GNOME desktop Unlike high-level languages such as Java and C++, assembly language is much closer to the machine code that actually runs computers; it's used to create programs or modules that are very fast and efficient, as well as in hacking exploits and reverse engineering Covering assembly language in the Pentium microprocessor environment, this code-intensive guide shows programmers how to create stand-alone assembly language programs as well as how to incorporate assembly language libraries or routines into existing high-level applications Demonstrates how to manipulate data, incorporate advanced functions and libraries, and maximize application performance Examples use C as a high-level language, Linux as the development environment, and GNU tools for assembling, compiling, linking, and debugging

Beginning Linux Programming, Fourth Edition continues its unique approach to teaching UNIX programming in a simple and structured way on the Linux platform. Through the use of detailed and realistic examples, students learn by doing, and are able to move from being a Linux beginner to creating custom applications in Linux. The book introduces fundamental concepts beginning with the basics of writing Unix programs in C, and including material on basic system calls, file I/O, interprocess communication (for getting programs to work together), and shell programming. Parallel to this, the book introduces the toolkits and libraries for working with user interfaces, from simpler terminal mode applications to X and GTK+ for graphical user interfaces. Advanced topics are covered in detail such as processes, pipes, semaphores, socket programming, using MySQL, writing applications for the GNOME or the KDE desktop, writing device drivers, POSIX Threads, and kernel programming for the latest Linux Kernel.

Written by a bestselling and well-known author, this is the only book on programming for Linux using GNU C++, covering all aspects of Linux including fundamentals, object-oriented programming, advanced techniques, X Windows, and more. CD contains Red Hat Linux source code and all the code from the text.

A detailed introduction to the C programming language for experienced programmers. The world runs on code written in the C programming language, yet most schools begin the curriculum with Python or Java. Effective C bridges this gap and brings C into the modern era--covering the modern C17 Standard as well as potential C2x features. With the aid of this instant classic, you'll soon be writing professional, portable, and secure C programs to power robust systems and solve real-world problems. Robert C. Seacord introduces C and the C Standard Library while addressing best practices, common errors, and open debates in the C community. Developed together with other C Standards committee experts, Effective C will teach you how to debug, test, and analyze C programs. You'll benefit from

Seacord's concise explanations of C language constructs and behaviors, and from his 40 years of coding experience. You'll learn:

- How to identify and handle undefined behavior in a C program
- The range and representations of integers and floating-point values
- How dynamic memory allocation works and how to use nonstandard functions
- How to use character encodings and types
- How to perform I/O with terminals and filesystems using C Standard streams and POSIX file descriptors
- How to understand the C compiler's translation phases and the role of the preprocessor
- How to test, debug, and analyze C programs

Effective C will teach you how to write professional, secure, and portable C code that will stand the test of time and help strengthen the foundation of the computing world.

The bash shell is a complete programming language, not merely a glue to combine external Linux commands. By taking full advantage of shell internals, shell programs can perform as snappily as utilities written in C or other compiled languages. And you will see how, without assuming Unix lore, you can write professional bash 4.0 programs through standard programming techniques. Complete bash coverage Teaches bash as a programming language Helps you master bash 4.0 features

The pressure is on during the interview process but with the right preparation, you can walk away with your dream job. This classic book uncovers what interviews are really like at America's top software and computer companies and provides you with the tools to succeed in any situation. The authors take you step-by-step through new problems and complex brainteasers they were asked during recent technical interviews. 50 interview scenarios are presented along with in-depth analysis of the possible solutions. The problem-solving process is clearly illustrated so you'll be able to easily apply what you've learned during crunch time. You'll also find expert tips on what questions to ask, how to approach a problem, and how to recover if you become stuck. All of this will help you ace the interview and get the job you want. What you will learn from this book

- Tips for effectively completing the job application
- Ways to prepare for the entire programming interview process
- How to find the kind of programming job that fits you best
- Strategies for choosing a solution and what your approach says about you
- How to improve your interviewing skills so that you can respond to any question or situation
- Techniques for solving knowledge-based problems, logic puzzles, and programming problems

Who this book is for This book is for programmers and developers applying for jobs in the software industry or in IT departments of major corporations. Wrox Beginning guides are crafted to make learning programming languages and technologies easier than you think, providing a structured, tutorial format that will guide you through all the techniques involved.

Linux for Developers shows you how to start writing great code for Linux, whether you're a Linux user with little or no coding experience, or an experienced Windows programmer. Leading IT trainer/author William "Bo" Rothwell begins with a clear and up-to-date review of modern open source software, including the licensing arrangements and tradeoffs all developers need to understand. He presents essential skills for both Linux command line and GUI environments, introducing text editors and other tools for efficient coding. Building on this knowledge, Rothwell introduces scripting tools such as Bash, Python, and Perl, as well as traditional object-oriented programming languages such as Java, C++, and C. Finally, he presents a full section on the powerful Git version control system, teaching skills you can use in Linux and many other environments. Access Linux systems, use GUIs, and work at the command line Learn how Linux organizes files and navigate its filesystem Use basic developer commands such as gzip and grep Edit programs with vi and vim, and explore alternative editors Perform basic sysadmin tasks that developers often need to handle Compare Linux languages to choose the best one for each task Write Bash scripts that interact with users or other shell features Program with Python and Perl: flow control, variables, and more Understand Linux features related to building C, C++, and Java programs Stay on top of complex projects with GIT revision control Work in GIT: staging, committing, branches, diffs, merges, and patches Manage local and remote GIT repositories This guide's modular coverage helps you quickly access whatever information you need right now.

\*The most updated PostgreSQL book on the market, covering version 8.0 \*Highlights the most popular PostgreSQL APIs, including C, Perl, PHP, and Java \*This is two books in one; it simultaneously covers key relational database design principles, while teaching PostgreSQL

\* Clear and abundant examples, using real-world code, written by three experienced developers who write networking code for a living. \* Describes how to build clients and servers, explains how TCP, UDP, and IP work, and shows how to debug networking applications via packet sniffing and deconstruction. \* Well suited for Windows developer looking to expand to Linux, or for the proficient Linux developer looking to incorporate client-server programming into their application.

Implement a SOHO or SMB Linux infrastructure to expand your business and associated IT capabilities. Backed by the expertise and experienced guidance of the authors, this book provides everything you need to move your business forward. Pro Linux System Administration makes it easy for small- to medium-sized businesses to enter the world of zero-cost software running on Linux and covers all the distros you might want to use, including Red Hat, Ubuntu, Debian, and CentOS. Pro Linux System Administration takes a layered, component-based approach to open source business systems, while training system administrators as the builders of business infrastructure. Completely updated for this second edition, Dennis Matotek takes you through an infrastructure-as-code approach, seamlessly taking you through steps along the journey of Linux administration with all you need to master complex systems. This edition now includes Jenkins, Ansible, Logstash and more. What You'll Learn: Understand Linux architecture Build, back up, and recover Linux servers Create basic networks and network services with Linux Build and implement Linux infrastructure and services including mail, web, databases, and file and print Implement Linux security Resolve Linux performance and capacity planning issues Who This Book Is For: Small to medium-sized business owners looking to run their own IT, system administrators considering migrating to Linux, and IT systems integrators looking for an extensible Linux infrastructure management approach.

This is the eBook version of the printed book. If the print book includes a CD-ROM, this content is not included within the eBook version. Advanced Linux Programming is divided into two parts. The first covers generic UNIX system services, but with a particular eye towards Linux specific information. This portion of the book will be of use even to advanced programmers who have worked with other Linux systems since it will cover Linux specific details and differences. For programmers without UNIX experience, it will be even more valuable. The second section covers material that is entirely Linux specific. These are truly advanced topics, and are the techniques that the gurus use to build great applications. While this book will focus mostly on the Application Programming Interface (API) provided by the Linux kernel and the C library, a preliminary introduction to the development tools available will allow all who purchase the book to make immediate use of Linux.

Linux is a free, UNIX-like operating system created by Linus Torvald in 1991 that is developed by a loosely knit team of programmers working from all over the world. It works from on almost every kind of computer in existence, and provides a robust platform for a wide variety of applications.

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.

Pro Bash Programming teaches you how to effectively utilize the Bash shell in your programming. The Bash shell is a complete programming language, not merely a glue to combine external Linux commands. By taking full advantage of Shell internals, Shell programs can perform as snappily as utilities written in C or other compiled languages. And you will see how, without assuming Unix lore, you can write professional Bash 4.3 programs through standard programming techniques. This second edition has updated for Bash 4.3, and many scripts have been rewritten to make them more idiomatically Bash, taking better advantage of features specific to Bash. It is easy to read, understand, and will teach you how to get to grips with Bash programming without drowning you in pages and pages of syntax. Using this book you will be able to use the shell efficiently, make scripts run faster using expansion and external commands, and understand how to overcome many common mistakes that cause scripts to fail. This book is perfect for all beginning Linux and Unix system administrators who want to be in full control of their systems, and really get to grips with Bash programming.

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.

Program in assembly starting with simple and basic programs, all the way up to AVX programming. By the end of this book, you will be able to write and read assembly code, mix assembly with higher level languages, know what AVX is, and a lot more than that. The code used in Beginning x64 Assembly Programming is kept as simple as possible, which means: no graphical user interfaces or whistles and bells or error checking. Adding all these nice features would distract your attention from the purpose: learning assembly language. The theory is limited to a strict minimum: a little bit on binary numbers, a short presentation of logical operators, and some limited linear algebra. And we stay far away from doing floating point conversions. The assembly code is presented in complete programs, so that you can test them on your computer, play with them, change them, break them. This book will also show you what tools can be used, how to use them, and the potential problems in those tools. It is not the intention to give you a comprehensive course on all of the assembly instructions, which is impossible in one book: look at the size of the Intel Manuals. Instead, the author will give you a taste of the main items, so that you will have an idea about what is going on. If you work through this book, you will acquire the knowledge to investigate certain domains more in detail on your own. The majority of the book is dedicated to assembly on Linux, because it is the easiest platform to learn assembly language. At the end the author provides a number of chapters to get you on your way with assembly on Windows. You will see that once you have Linux assembly under your belt, it is much easier to take on Windows assembly. This book should not be the first book you read on programming, if you have never programmed before, put this book aside for a while and learn some basics of programming with a higher-level language such as C. What You Will Learn Discover how a CPU and memory works Appreciate how a computer and operating system work together See how high-level language compilers generate machine language, and use that knowledge to write more efficient code Be better equipped to analyze bugs in your programs Get your program working, which is the fun part Investigate malware and take the necessary actions and precautions Who This Book Is For Programmers in high level languages. It is also for systems engineers and security engineers working for malware investigators. Required knowledge: Linux, Windows, virtualization, and higher level programming languages (preferably C or C++).

There's a lot to be said for going back to basics. Not only does this Bible give you a quick refresher on the structure of open-source Linux software, it also shows you how to bypass the hefty graphical user interface on Linux systems and start interacting the fast and efficient way?with command lines and automated scripts. You'll learn how to manage files on the filesystem, start and stop programs, use databases, even do Web programming?without a GUI?with this one-stop resource.

For software developers, it's the holy grail: write one state-of-the-art graphical application that runs on Linux, UNIX, and Windows. Qt 2 Programming for Linux and Windows shows experienced C++ programmers how to do just that, using the powerful new Qt 2.x toolkits -- the same tools used to build the #1 Linux graphical user interface, KDE.

A number of widely used contemporary processors have instruction-set extensions for improved performance in multi-media applications. The aim is to allow operations to proceed on multiple pixels each clock cycle. Such instruction-sets have been incorporated both in specialist DSPchips such as the Texas C62xx (Texas Instruments, 1998) and in general purpose CPU chips like the Intel IA32 (Intel, 2000) or the AMD K6 (Advanced Micro Devices, 1999). These instruction-set extensions are typically based on the Single Instruction-stream Multiple Data-stream (SIMD) model in which a single instruction causes the same mathematical operation to be carried out on several operands, or pairs of operands, at the same time. The level of parallelism supported ranges from two floating point operations, at a time on the AMD K6 architecture to 16 byte operations at a time on the Intel P4 architecture. Whereas processor architectures are moving

towards greater levels of parallelism, the most widely used programming languages such as C, Java and Delphi are structured around a model of computation in which operations take place on a single value at a time. This was appropriate when processors worked this way, but has become an impediment to programmers seeking to make use of the performance offered by multi-media instruction -sets. The introduction of SIMD instruction sets (Peleg et al.

Shell Programming in Unix, Linux and OS X is a thoroughly updated revision of Kochan and Wood's classic Unix Shell Programming tutorial. Following the methodology of the original text, the book focuses on the POSIX standard shell, and teaches you how to develop programs in this useful programming environment, taking full advantage of the underlying power of Unix and Unix-like operating systems. After a quick review of Unix utilities, the book's authors take you step-by-step through the process of building shell scripts, debugging them, and understanding how they work within the shell's environment. All major features of the shell are covered, and the large number of practical examples make it easy for you to build shell scripts for your particular applications. The book also describes the major features of the Korn and Bash shells. Learn how to... Take advantage of the many utilities provided in the Unix system Write powerful shell scripts Use the shell's built-in decision-making and looping constructs Use the shell's powerful quoting mechanisms Make the most of the shell's built-in history and command editing capabilities Use regular expressions with Unix commands Take advantage of the special features of the Korn and Bash shells Identify the major differences between versions of the shell language Customize the way your Unix system responds to you Set up your shell environment Make use of functions Debug scripts Contents at a Glance 1 A Quick Review of the Basics 2 What Is the Shell? 3 Tools of the Trade 4 And Away We Go 5 Can I Quote You on That? 6 Passing Arguments 7 Decisions, Decisions 8 'Round and 'Round She Goes 9 Reading and Printing Data 10 Your Environment 11 More on Parameters 12 Loose Ends 13 Rolo Revisited 14 Interactive and Nonstandard Shell Features A Shell Summary B For More Information 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.

Bestselling UNIX author Stevens offers application and system programmers his professional, experienced-based guidance on using the system call interface with C. Since good examples are the key to a book like this, a simple shell program is developed in the first chapter and then expanded throughout the book to demonstrate the principles.

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: Embedded Linux runs many of the devices we use every day. From smart TVs and Wi-Fi routers to test equipment and industrial controllers, all of them have Linux at their heart. The Linux OS is one of the foundational technologies comprising the core of the Internet of Things (IoT). This book starts by breaking 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 explains 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 runit 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 Linux implementation on embedded devices, then this book is for you. Embedded systems engineers accustomed to programming for low-power microcontrollers can use this book to help make the leap to high-speed systems on chips that can run Linux. Anyone responsible for developing new hardware that needs to run Linux will also find this book useful. Basic working knowledge of the POSIX standard, C programming, and shell scripting is assumed.

The Art of UNIX Programming poses the belief that understanding the unwritten UNIX engineering tradition and mastering its design patterns will help programmers of all stripes to become better programmers. This book attempts to capture the engineering wisdom and design philosophy of the UNIX, Linux, and Open Source software development community as it has evolved over the past three decades, and as it is applied today by the most experienced programmers. Eric Raymond offers the next generation of "hackers" the unique opportunity to learn the connection between UNIX philosophy and practice through careful case studies of the very best UNIX/Linux programs.

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.

Linux for Developers shows you how to start writing great code for Linux, whether you're a Linux user with little or no coding experience, or an experienced Windows programmer. Leading IT trainer/author William "Bo" Rothwell begins with a clear and up-to-date review of modern open source software, including the licensing arrangements and tradeoffs all developers need to understand. He presents essential skills for both Linux command line and GUI environments, introducing text editors and other tools for efficient coding. Building on this knowledge, Rothwell introduces scripting tools such as Bash, Python, and Perl, as well as traditional object-oriented programming languages such as Java, C++, and C. Finally, he presents a full section on the powerful Git version control system, teaching skills you can use in Linux and many other environments. Access Linux systems, use GUIs, and work at the command line Learn how Linux

organizes files and navigate its filesystem Use basic developer commands such as gzip and grep Edit programs with vi and vim, and explore alternative editors Perform basic sysadmin tasks that developers often need to handle Compare Linux languages to choose the best one for each task Write Bash scripts that interact with users or other shell features Program with Python and Perl: flow control, variables, and more Understand Linux features related to building C, C++, and Java programs Stay on top of complex projects with GIT revision control Work in GIT: staging, committing, branches, diffs, merges, and patches Manage local and remote GIT repositories This guide's modular coverage helps you quickly access whatever information you need right now.

The revision of the definitive guide to Unix system programming is now available in a more portable format.

Master the techniques needed to build great, efficient embedded devices on Linux About This Book Discover how to build and configure reliable embedded Linux devices This book has been updated to include Linux 4.9 and Yocto Project 2.2 (Morty) This comprehensive guide covers the remote update of devices in the field and power management Who This Book Is For If you are an engineer who wishes to understand and use Linux in embedded devices, this book is for you. It is also for Linux developers and system programmers who are familiar with embedded systems and want to learn and program the best in class devices. It is appropriate for students studying embedded techniques, for developers implementing embedded Linux devices, and engineers supporting existing Linux devices. What You Will Learn Evaluate the Board Support Packages offered by most manufacturers of a system on chip or embedded module Use Buildroot and the Yocto Project to create embedded Linux systems quickly and efficiently Update IoT devices in the field without compromising security Reduce the power budget of devices to make batteries last longer Interact with the hardware without having to write kernel device drivers Debug devices remotely using GDB, and see how to measure the performance of the systems using powerful tools such as perk, ftrace, and valgrind Find out how to configure Linux as a real-time operating system In Detail Embedded Linux runs many of the devices we use every day, from smart TVs to WiFi routers, test equipment to industrial controllers - all of them have Linux at their heart. Linux is a core technology in the implementation of the inter-connected world of the Internet of Things. The comprehensive guide shows you the technologies and techniques required to build Linux into embedded systems. You will begin by learning about the fundamental elements that underpin all embedded Linux projects: the toolchain, the bootloader, the kernel, and the root filesystem. You'll see how to create each of these elements from scratch, and how to automate the process using Buildroot and the Yocto Project. Moving on, you'll find out how to implement an effective storage strategy for flash memory chips, and how to install updates to the device remotely once it is deployed. You'll also get to know the key aspects of writing code for embedded Linux, such as how to access hardware from applications, the implications of writing multi-threaded code, and techniques to manage memory in an efficient way. The final chapters show you how to debug your code, both in applications and in the Linux kernel, and how to profile the system so that you can look out for performance bottlenecks. By the end of the book, you will have a complete overview of the steps required to create a successful embedded Linux system. Style and approach This book is an easy-to-follow and pragmatic guide with in-depth analysis of the implementation of embedded devices. It follows the life cycle of a project from inception through to completion, at each stage giving both the theory that underlies the topic and practical step-by-step walkthroughs of an example implementation. The Linux Programming Interface (TLPI) is the definitive guide to the Linux and UNIX programming interface—the interface employed by nearly every application that runs on a Linux or UNIX system. In this authoritative work, Linux programming expert Michael Kerrisk provides detailed descriptions of the system calls and library functions that you need in order to master the craft of system programming, and accompanies his explanations with clear, complete example programs. You'll find descriptions of over 500 system calls and library functions, and more than 200 example programs, 88 tables, and 115 diagrams. You'll learn how to: –Read and write files efficiently –Use signals, clocks, and timers –Create processes and execute programs –Write secure programs –Write multithreaded programs using POSIX threads –Build and use shared libraries –Perform interprocess communication using pipes, message queues, shared memory, and semaphores –Write network applications with the sockets API While The Linux Programming Interface covers a wealth of Linux-specific features, including epoll, inotify, and the /proc file system, its emphasis on UNIX standards (POSIX.1-2001/SUSv3 and POSIX.1-2008/SUSv4) makes it equally valuable to programmers working on other UNIX platforms. The Linux Programming Interface is the most comprehensive single-volume work on the Linux and UNIX programming interface, and a book that's destined to become a new classic.

[Copyright: 2efc92f650e27c0cfa6b8809909923e4](#)