

## Linux System Programming: Talking Directly To The Kernel And C Library

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 perf, 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.

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

Authoried by two of the leading authorities in the field, this guide offers readers the knowledge and skills needed to achieve proficiency with embedded software.

This is Linux for those of us who don't mind typing. All Linux users and administrators tend to like the flexibility and speed of Linux administration from the command line in byte-sized chunks, instead of fairly standard graphical user interfaces. Beginning the Linux Command Line is verified against all of the most important Linux distributions, and follows a task-oriented approach which is distribution agnostic. Now this Second Edition of Beginning the Linux Command Line updates to the very latest versions of the Linux Operating System, including the new Btrfs file system and its management, and systemd boot procedure and firewall management with firewallD! Updated to the latest versions of Linux Work with files and directories, including Btrfs! Administer users and security, and deploy firewallD Understand how Linux is organized, to think Linux!

The Art of UNIX Programming

Create fast and reliable embedded solutions with Linux 5.4 and the Yocto Project 3.1 (Dunfell)

Hands-On System Programming with Go

ABCs of z/OS System Programming:

Python for Unix and Linux System Administration

Advanced Programming in the UNIX Environment

Build, customize, and debug your own Android system About This Book Master Android system-level programming by integrating, customizing, and extending popular open source projects Use Android emulators to explore the true potential of your hardware Master key debugging techniques to create a hassle-free development environment Who This Book Is For This book is for Android system programmers and developers who want to use Android and create indigenous projects with it. You should know the important points about the operating system and the C/C++ programming language. What You Will Learn Set up the Android development environment and organize source code repositories Get acquainted with the Android system architecture Build the Android emulator from the AOSP source tree Find out how to enable WiFi in the Android emulator Debug the boot up process using a customized Ramdisk Port your Android system to a new platform using VirtualBox Find out what recovery is and see how to enable it in the AOSP build Prepare and test OTA packages In Detail Android system programming involves both hardware and software knowledge to work on system level programming. The developers need to use various techniques to debug the different components in the target devices. With all the challenges, you usually have a deep learning curve to master relevant knowledge in this area. This book will not only give you the key knowledge you need to understand Android system programming, but will also prepare you as you get hands-on with projects and gain debugging skills that you can use in your future projects. You will start by exploring the basic setup of AOSP, and building and testing an emulator image. In the first project, you will learn how to customize and extend the Android emulator. Then you'll move on to the real challenge—building your own Android system on VirtualBox. You'll see how to debug the init process, resolve the bootloader issue, and enable various hardware interfaces. When you have a complete system, you will learn how to patch and upgrade it through recovery. Throughout the book, you will get to know useful tips on how to integrate and reuse existing open source projects such as LineageOS (CyanogenMod), Android-x86, Xposed, and GApps in your own system. Style and approach This is an easy-to-follow guide full of hands-on examples and system-level programming tips.

Covering all the essential components of Unix/Linux, including process management, concurrent programming, timer and time service, file systems and network programming, this textbook emphasizes programming practice in the Unix/Linux environment. Systems Programming in Unix/Linux is intended as a textbook for systems programming courses in technically-oriented Computer Science/Engineering curricula that emphasize both theory and programming practice. The book contains many detailed working example programs with complete source code. It is also suitable for self-study by advanced programmers and computer enthusiasts. Systems programming is an indispensable part of Computer Science/Engineering education. After taking an introductory programming course, this book is meant to further knowledge by detailing how dynamic data structures are used in practice, using programming exercises and programming projects on such topics as C structures, pointers, link lists and trees. This book provides a wide range of knowledge about computer systemssoftware and advanced programming skills, allowing readers to interface with operatingsystem kernel, make efficient use of system resources and develop application software.It also prepares readers with the needed background to pursue advanced studies inComputer Science/Engineering, such as operating systems, embedded systems, databasesystems, data mining, artificial intelligence, computer networks, network security,distributed and parallel computing.

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.

You've experienced the shiny, point-and-click surface of your Linux computer—now dive below and explore its depths with the power of the command line. The Linux Command Line takes you from your very first terminal keystrokes to writing full programs in Bash, the most popular Linux shell. Along the way you'll learn the timeless skills handed down by generations of gray-bearded, mouse-shunning gurus: file navigation, environment configuration, command chaining, pattern matching with regular expressions, and more. In addition to that practical knowledge, author William Shotts reveals the philosophy behind these tools and the rich heritage that your desktop Linux machine has inherited from Unix supercomputers of yore. As you make your way through the book's short, easily-digestible chapters, you'll learn how to: \* Create and delete files, directories, and symlinks \* Administer your system, including networking, package installation, and process management \* Use standard input and output, redirection, and pipelines \* Edit files with Vi, the world's most popular text editor \* Write shell scripts to automate common or boring tasks \* Slice and dice text files with cut, paste, grep, patch, and sed Once you overcome your initial "shell shock," you'll find that the command line is a natural and expressive way to communicate with your computer. Just don't be surprised if your mouse starts to gather dust. A featured resource in the Linux Foundation's "Evolution of a SysAdmin"

Understanding Linux Network Internals

Learning the bash Shell

Linux System Programming Techniques

A Complete Introduction

Advanced Linux Programming

Building a Modern Computer from First Principles

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 filesystems 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.

This title gives students an integrated and rigorous picture of applied computer science, as it comes to play in the construction of a simple yet powerful computer system.

Explore the fundamentals of systems programming starting from kernel API and filesystem to network programming and process communications Key FeaturesLearn how to write Unix and Linux system code in Golang v1.12Perform inter-process communication using pipes, message queues, shared memory, and semaphoresExplore modern Go features such as goroutines and channels that facilitate systems programmingBook Description System software and applications were largely created using low-level languages such as C or C++. Go is a modern language that combines simplicity, concurrency, and performance, making it a good alternative for building system applications for Linux and macOS. This Go book introduces Unix and systems programming to help you understand the components the OS has to offer, ranging from the kernel API to the filesystem, and familiarize yourself with Go and its specifications. You'll also learn how to optimize input and output operations with files and streams of data, which are useful tools in building pseudo terminal applications. You'll gain insights into how processes communicate with each other, and learn about processes and daemon control using signals, pipes, and exit codes. This book will also enable you to understand how to use network communication using various protocols, including TCP and HTTP. As you advance, you'll focus on Go's best feature-concurrency helping you handle communication with channels and goroutines, other concurrency tools to synchronize shared resources, and the context package to write elegant applications. By the end of this book, you will have learned how to build concurrent system applications using Go What you will learnExplore concepts of system programming using Go and concurrencyGain insights into Golang's internals, memory models and allocationFamiliarize yourself with the filesystem and IO streams in generalHandle and control processes and daemons' lifetime via signals and pipesCommunicate with other applications effectively using a networkUse various encoding formats to serialize complex data structuresBecome well-versed in concurrency with channels, goroutines, and syncUse concurrency patterns to build robust and performant system applicationsWho this book is for If you are a developer who wants to learn system programming with Go, this book is for you. Although no knowledge of Unix and Linux system programming is necessary, intermediate knowledge of Go will help you understand the concepts covered in the book

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.

Systems Programming in Unix/Linux

Beginning Linux?Programming

Hands-On System Programming with Linux

UNIX Systems Programming

Mastering Embedded Linux Programming

Practical System Programming for Rust Developers

**bull; Learn UNIX essentials with a concentration on communication, concurrency, and multithreading techniques bull; Full of ideas on how to design and implement good software along with unique projects throughout bull; Excellent companion to Stevens' Advanced UNIX System Programming**

**Introduces the features of the C programming language, discusses data types, variables, operators, control flow, functions, pointers, arrays, and structures, and looks at the UNIX system interface**

**Harness the power of Linux to create versatile and robust embedded solutions Key FeaturesLearn how to develop and configure robust embedded Linux devicesExplore 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 kernelBook 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 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 learnUse Buildroot and the Yocto Project to create embedded Linux systemsTroubleshoot BitBake build failures and streamline your Yocto development workflowUpdate IoT devices securely in the field using Mender or balenaPrototype peripheral additions by reading schematics, modifying device trees, soldering breakout boards, and probing pins with a logic analyzerInteract with hardware without having to write kernel device driversDivide your system up into services supervised by BusyBox runitDebug devices remotely using GDB and measure the performance of systems using tools such as perf, ftrace, eBPF, and CallgrindWho 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.**

**Learning the new system's programming language for all Unix-type systems About This Book Learn how to write system's level code in Golang, similar to Unix/Linux systems code Ramp up in Go quickly Deep dive into Goroutines and Go concurrency to be able to take advantage of Go server-level constructs Who This Book Is For Intermediate Linux and general Unix programmers. Network programmers from beginners to advanced practitioners. C and C++ programmers interested in different approaches to concurrency and Linux systems programming. What You Will Learn Explore the Go language from the standpoint of a developer conversant with Unix, Linux, and so on Understand Goroutines, the lightweight threads used for systems and concurrent applications Learn how to translate Unix and Linux systems code in C to Golang code How to write fast and lightweight server code Dive into concurrency with Go Write low-level networking code In Detail Go is the new systems programming language for Linux and Unix systems. It is also the language in which some of the most prominent cloud-level systems have been written, such as Docker. Where C programmers used to rule, Go programmers are in demand to write highly optimized systems programming code. Created by some of the original designers of C and Unix, Go expands the systems programmers toolkit and adds a mature, clear programming language. Traditional system applications become easier to write since pointers are not relevant and garbage collection has taken away the most problematic area for low-level systems code: memory management. This book opens up the world of high-performance Unix system applications to the beginning Go programmer. It does not get stuck on single systems or even system types, but tries to expand the original teachings from Unix system level programming to all types of servers, the cloud, and the web. Style and approach This is the first book to introduce Linux and Unix systems programming in Go, a field for which Go has actually been developed in the first place.**

Linux Kernel Development

Linux Kernel Programming

Android System Programming

The Linux Programming Interface

Talking Directly to the Kernel and C Library

Go Systems Programming

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.

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.

Explore various Rust features, data structures, libraries, and toolchain to build modern systems software with the help of hands-on examples Key FeaturesLearn techniques to design and build system tools and utilities in RustExplore the different features of the Rust standard library for interacting with operating systemsGain an in-depth understanding of the Rust programming language by writing low-level softwareBook Description Modern programming languages such as Python, JavaScript, and Java have become increasingly accepted for application-level programming, but for systems programming, C and C++ are predominantly used due to the need for low-level control of system resources. Rust promises the best of both worlds: the type safety of Java, and the speed and expressiveness of C++, while also including memory safety without a garbage collector. This book is a comprehensive introduction if you're new to Rust and systems programming and are looking to build reliable and efficient systems software without C or C++. The book takes a unique approach by starting each topic with Linux kernel concepts and APIs relevant to that topic. You'll also explore how system resources can be controlled from Rust. As you progress, you'll delve into advanced topics. You'll cover network programming, focusing on aspects such as working with low-level network primitives and protocols in Rust, before going on to learn how to use and compile Rust with WebAssembly. Later chapters will take you through practical code examples and projects to help you build on your knowledge. By the end of this Rust programming book, you will be equipped with practical skills to write systems software tools, libraries, and utilities in Rust. What you will learnGain a solid understanding of how system resources are managedUse Rust confidently to control and operate a Linux or Unix systemUnderstand how to write a host of practical systems software tools and utilitiesDelve into memory management with the memory layout of Rust programsDiscover the capabilities and features of the Rust Standard LibraryExplore external crates to improve productivity for future Rust programming projectsWho this book is for This book is for developers with basic knowledge of Rust but little to no knowledge or experience of systems programming. System programmers who want to consider Rust as an alternative to C or C++ will also find this book useful.

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.

Unix Shell Programming

A Linux and UNIX System Programming Handbook

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

Beginning the Linux Command Line

An Introduction to Professional C Programming

Build fast and secure software for Linux/Unix systems with the help of practical examples

**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 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.

**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. You'll take an in-depth look at Linux from both a theoretical and an applied perspective over a wide range of programming topics, including: 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 you have, 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 **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.

**Python is an ideal language for solving problems, especially in Linux and Unix networks.** With this pragmatic book, administrators can review various tasks that often occur in the management of these systems, and learn how Python can provide a more efficient and less painful way to handle them. Each chapter in Python for Unix and Linux System Administration presents a particular administrative issue, such as concurrency or data backup, and presents Python solutions through hands-on examples. Once you finish this book, you'll be able to develop your own set of command-line utilities with Python to tackle a wide range of problems. Discover how this language can help you: Read text files and extract information Run tasks concurrently using the threading and forking options Get information from one process to another using network facilities Create clickable GUIs to handle large and complex utilities Monitor large clusters of machines by interacting with SNMP programmatically Master the IPython Interactive Python shell to replace or augment Bash, Korn, or Z-Shell Integrate Cloud Computing into your infrastructure, and learn to write a Google App Engine Application Solve unique data backup challenges with customized scripts Interact with MySQL, SQLite, Oracle, Postgres, Django ORM, and SQLAlchemy With this book, you'll learn how to package and deploy your Python applications and libraries, and write code that runs equally well on multiple Unix platforms. You'll also learn about several Python-related technologies that will make your life much easier.

The Elements of Computing Systems

The Linux Command Line

Master Linux and Unix system level programming with Go

Explore Linux system programming interfaces, theory, and practice

Effective C

Understanding the Linux Kernel

**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.

**Get up and running with system programming concepts in Linux** Key FeaturesAcquire insight on Linux system architecture and its programming interfacesGet to grips with core concepts such as process management, signalling and pthreadsPacked with industry best practices and dozens of code examplesBook Description The Linux OS and its embedded and server applications are critical components of today's software infrastructure in a decentralized, networked universe. The industry's demand for proficient Linux developers is only rising with time. Hands-On System Programming with Linux gives you a solid theoretical base and practical industry-relevant descriptions, and covers the Linux system programming domain. It delves into the art and science of Linux application programming-- system architecture, process memory and management, signaling, timers, pthreads, and file IO. This book goes beyond the use API X to do Y approach; it explains the concepts and theories required to understand programming interfaces and design decisions, the tradeoffs made by experienced developers when using them, and the rationale behind them. Troubleshooting tips and techniques are included in the concluding chapter. By the end of this book, you will have gained essential conceptual design knowledge and hands-on experience working with Linux system programming interfaces. What you will learnExplore the theoretical underpinnings of Linux system architectureUnderstand why modern OSes use virtual memory and dynamic memory APIsGet to grips with dynamic memory issues and effectively debug themLearn key concepts and powerful system APIs related to process managementEffectively perform file IO and use signaling and timersDeeply understand multithreading concepts, pthreads APIs, synchronization and schedulingWho this book is for Hands-On System Programming with Linux is for Linux system engineers, programmers, or anyone who wants to go beyond using an API set to understanding the theoretical underpinnings and concepts behind powerful Linux system programming APIs. To get the most out of this book, you should be familiar with Linux at the user-level logging in, using shell via the command line interface, the ability to use tools such as find, grep, and sort. Working knowledge of the C programming language is required. No prior experience with Linux systems programming is assumed.

**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.**

**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.

**Become a proficient Linux system programmer using expert recipes and techniques**

**Building Embedded Linux Systems**

**Linux System Programming**

**Embedded Linux System Design and Development**

**Linux for Developers**

**Communication, Concurrency, and Threads**

This IBM® Redbooks® publication describes the functions of z/OS® Communications Server. z/OS Communications Server provides a set of communications protocols that support peer-to-peer connectivity functions for both local and wide-area networks, including the most popular wide-area network, the Internet. z/OS Communications Server also provides performance enhancements that can benefit a variety of TCP/IP applications. z/OS Communications Server provides both SNA and TCP/IP networking protocols for z/OS. The SNA protocols are provided by VTAM® and include Subarea, Advanced Peer-to-Peer Networking, and High Performance Routing protocols. z/OS Communications Server exploits z/OS UNIX® services even for traditional MVSTM environments and applications. Prior to utilizing TCP/IP services, therefore, a full-function mode z/OS UNIX environment including a Data Facility Storage Management Subsystem (DFSMSdfp), a z/OS UNIX file system, and a security product (such as Resource Access Control Facility, or RACF®) must be defined and active before z/OS Communications Server can be started successfully. The ABCs of z/OS System Programming is a 13-volume collection that provides an introduction to the z/OS operating system and the hardware architecture. Whether you are a beginner or an experienced system programmer, the ABCs collection provides the information that you need to start your research into z/OS and related subjects. If you want to become more familiar with z/OS in your current environment, or if you are evaluating platforms to consolidate your e-business applications, the ABCs collection will serve as a powerful technical tool. The contents of the volumes are as follows: Volume 1: Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation Volume 2: z/OS implementation and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKLST, authorized libraries, SMP/E, Language Environment® Volume 3: Introduction to DFSMS, data set basics storage management hardware and software, catalogs, and DFSMSStvs Volume 4: Communication Server, TCP/IP, and VTAM Volume 5: Base and Parallel Sysplex®, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, automatic restart management (ARM), Geographically Dispersed Parallel Sysplex™ (GDPS®) Volume 6: Introduction to security, RACF, Digital certificates and PKI, Kerberos, cryptography and z990 integrated cryptography, zSeries® firewall technologies, LDAP, and Enterprise identity mapping (EIM) Volume 7: Printing in a z/OS environment, Infoprint Server and Infoprint Central Volume 8: An introduction to z/OS problem diagnosis Volume 9: z/OS UNIX System Services Volume 10: Introduction to z/Architecture®, zSeries processor design, zSeries connectivity, LPAR concepts, HCD, and HMC Volume 11: Capacity planning, performance management, RMFTM, and SMF Volume 12: WLM Volume 13: JES3

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 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.

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.

The C Programming Language

Implementation and Theory

With C and GNU Development Tools

Linux Device Drivers

Build modern and concurrent applications for Unix and Linux systems using Golang

Programming Embedded Systems



Based upon the authors' experience in designing and deploying an embedded Linux system with a variety of applications, Embedded Linux System Design and Development contains a full embedded Linux system development roadmap for systems architects and software programmers. Explaining the issues that arise out of the use of Linux in embedded systems, the book facilitates movement to embedded Linux from traditional real-time operating systems, and describes the system design model containing embedded Linux. This book delivers practical solutions for writing, debugging, and profiling applications and drivers in embedded Linux, and for understanding Linux BSP architecture. It enables you to understand: various drivers such as serial, I2C and USB gadgets; uClinux architecture and its programming model; and the embedded Linux graphics subsystem. The text also promotes learning of methods to reduce system boot time, optimize memory and storage, and find memory leaks and corruption in applications. This volume benefits IT managers in planning to choose an embedded Linux distribution and in creating a roadmap for OS transition. It also describes the application of the Linux licensing model in commercial products.

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.

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)

O'Reilly's bestselling book on Linux's bash shell is at it again. Now that Linux is an established player both as a server and on the desktop Learning the bash Shell has been updated and refreshed to account for all the latest changes. Indeed, this third edition serves as the most valuable guide yet to the bash shell.As any good programmer knows, the first thing users of the Linux operating system come face to face with is the shell the UNIX term for a user interface to the system. In other words, it's what lets you communicate with the computer via the keyboard and display. Mastering the bash shell might sound fairly simple but it isn't. In truth, there are many complexities that need careful explanation, which is just what Learning the bash Shell provides.If you are new to shell programming, the book provides an excellent introduction, covering everything from the most basic to the most advanced features. And if you've been writing shell scripts for years, it offers a great way to find out what the new shell offers. Learning the bash Shell is also full of practical examples of shell commands and programs that will make everyday use of Linux that much easier. With this book, programmers will learn: How to install bash as your login shell The basics of interactive shell use, including UNIX file and directory structures, standard I/O, and background jobs Command line editing, history substitution, and key bindings How to customize your shell environment without programming The nuts and bolts of basic shell programming, flow control structures, command-line options and typed variables Process handling, from job control to processes, coroutines and subshells Debugging techniques, such as trace and verbose modes Techniques for implementing system-wide shell customization and features related to system security

Linux in a Nutshell

Linux Kernel Networking

Jumpstart Your Linux Programming Skills

The Linux Kernel Module Programming Guide

Beginning Linux Programming

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

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, strace, and gdb are among the packages discussed.

Find solutions to all your problems related to Linux system programming using practical recipes for developing your own system programs Key FeaturesDevelop a deeper understanding of how Linux system programming worksGain hands-on experience of working with different Linux projects with the help of practical examplesLearn how to develop your own programs for LinuxBook Description Linux is the world's most popular open source operating system (OS). Linux System Programming Techniques will enable you to extend the Linux OS with your own system programs and communicate with other programs on the system. The book begins by exploring the Linux filesystem, its basic commands, built-in manual pages, the GNU compiler collection (GCC), and Linux system calls. You'll then discover how to handle errors in your programs and will learn to catch errors and print relevant information about them. The book takes you through multiple recipes on how to read and write files on the system, using both streams and file descriptors. As you advance, you'll delve into forking, creating zombie processes, and daemons, along with recipes on how to handle daemons using systemd. After this, you'll find out how to create shared libraries and start exploring different types of interprocess communication (IPC). In the later chapters, recipes on how to write programs using POSIX threads and how to debug your programs using the GNU debugger (GDB) and Valgrind will also be covered. By the end of this Linux book, you will be able to develop your own system programs for Linux, including daemons, tools, clients, and filters. What you will learnDiscover how to write programs for the Linux system using a wide variety of system callsDelve into the working of POSIX functionsUnderstand and use key concepts such as signals, pipes, IPC, and process managementFind out how to integrate programs with a Linux systemExplore advanced topics such as filesystem operations, creating shared libraries, and debugging your programsGain an overall understanding of how to debug your programs using ValgrindWho this book is for This book is for anyone who wants to develop system programs for Linux and gain a deeper understanding of the Linux system. The book is beneficial for anyone who is facing issues related to a particular part of Linux system programming and is looking for specific recipes or solutions.