

System Programming With C And Unix 1st Edition Free

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.

This book constitutes the proceedings of the 19th Asian Symposium on Programming Languages and Systems, APLAS 2021, held in Chicago, USA, in October 2021.* The 17 papers presented in this volume were carefully reviewed and selected from 43 submissions. They were organized in topical sections named: analysis and synthesis, compilation and transformation, language, and verification. * The conference was held in a hybrid format due to the COVID-19 pandemic.

A revolutionary approach to enhancing productivity, creating flow, and vastly increasing your ability to capture, remember, and benefit from the unprecedented amount of information all around us. For the first time in history, we have instantaneous access to the world's knowledge. There has never been a better time to learn, to contribute, and to improve ourselves. Yet, rather than feeling empowered, we are often left feeling overwhelmed by this constant influx of information. The very knowledge that was supposed to set us free has instead led to the paralyzing stress of believing we'll never know or remember enough. Now, this eye-opening and accessible guide shows how you can easily create your own personal system for knowledge management, otherwise known as a Second Brain. As a trusted and organized digital repository of your most valued ideas, notes, and creative work synced across all your devices and platforms, a Second Brain gives you the confidence to tackle your most important projects and ambitious goals. Discover the full potential of your ideas and translate what you know into more powerful, more meaningful improvements in your work and life by Building a Second Brain.

Learn to write advanced C programs that are strongly type-checked, compact, and easy to maintain. This book focuses on real-life applications and problem solving in networking, database development, compilers, operating systems, and CAD.

The Linux Programming Interface

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

Android System Programming

Build performant and concurrent Unix and Linux systems with C++17

A Guide to System Programming

Explore the fundamentals of systems programming starting from kernel API and filesystem to network programming and process communications **Key Features**Learn 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 programming guide explains concepts, basic techniques, and common problems related to embedded systems software development. It features source code templates that can be used and reused in developing embedded software. Source code examples are included for both Intel and Motorola systems on a 3.5-inch diskette.

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

With this comprehensive text, Solaris practitioners will find all the information they need as they face and overcome significant challenges of their everyday work. Real-world case studies, poignant examples, and illustrative diagrams are rolled into this thorough reference.

C Programming For the PC the MAC and the Arduino Microcontroller System

A Complete Guide to Programming in C++

Solaris Systems Programming

Linux System Programming Techniques

Hands-On System Programming with Go

Systems Programming

This book teaches systems programming with the latest versions of C through a set of practical examples and problems. It covers the development of a handful of programs, implementing efficient coding examples. Practical Systems Programming with C contains three main parts: getting your hands dirty with C programming; practical systems programming using concepts such as processes, signals, and inter-process communication; and advanced socket-based programming which consists of developing a network application for reliable communication. You will be introduced to a marvelous ecosystem of systems programming with C, from handling basic system utility commands to communicating through socket programming. With the help of socket programming you will be able to build client-server applications in no time. The [secret sauce] of this book is its curated list of topics and solutions, which fit together through a set of different pragmatic examples; each topic is covered from scratch in an easy-to-learn way. On that journey, you'll focus on practical implementations and an outline of best practices and potential pitfalls. The book also includes a bonus chapter with a list of advanced topics and directions to grow your skills. What You Will Learn Program with operating systems using the latest version of C Work with Linux Carry out multithreading with C Examine the POSIX standard Work with files, directories, processes, and signals Explore IPC and how to work with it Who This Book Is For Programmers who have an exposure to C programming and want to learn systems programming. This book will help them to learn about core concepts of operating systems with the help of C programming. .

This guide was written for readers interested in learning the C++ programming language from scratch, and for both novice and advanced C++ programmers wishing to enhance their knowledge of C++. The text is organized to guide the reader from elementary language concepts to professional software development, with in depth coverage of all the C++ language elements en route.

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

An introduction to embedding systems for C and C++++ programmers encompasses such topics as testing memory devices, writing and erasing Flash memory, verifying nonvolatile memory contents, and much more. Original. (Intermediate).

COMPUTER SYSTEM AND PROGRAMMING IN C

19th Asian Symposium, APLAS 2021, Chicago, IL, USA, October 17-18, 2021, Proceedings

Learn socket programming in C and write secure and optimized network code

Explore Linux system programming interfaces, theory, and practice

Pragmatic Example Applications in Linux and Other Operating Systems

Embedded Systems Programming in C and Assembly

Delve into programming the Windows operating system through the Windows API in with C++. Use the power of the Windows API to working with processes, threads, jobs, memory, I/O and more. The book covers current Windows 10 versions, allowing you to get the most of what Windows has to offer to developers in terms of productivity, performance and scalability.

Explore various Rust features, data structures, libraries, and toolchain to build modern systems software with the help of hands-on examples **Key Features**Learn 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.

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.

This text is an introduction to the design and implementation of various types of system software. A central theme of the book is the relationship between machine architecture and systems software. The third edition has been updated to include current architecture, and the coverage of Operating Systems now includes shared/distributed memory and client/server systems. This book contains a wide selection of examples and exercises which are all optional, providing flexibility to instructors by allowing them to concentrate on the software and architecture they want to cover.--Publisher website.

Programming Languages and Systems

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

Rust in Action

UNIX System Programming Using C++

Master Linux and Unix system level programming with Go

Introduction to Computer Architecture and Systems Programming

Authored 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 book teaches system programming with the latest versions of C through a set of practical examples and problems. It covers the development of a handful of programs, implementing efficient coding examples. Practical System Programming with C contains three main parts: getting your hands dirty with multithreaded C programming; practical system programming using concepts such as processes, signals, and inter-process communication; and advanced socket-based programming which consists of developing a network application for reliable communication. You will be introduced to a marvelous ecosystem of system programming with C, from handling basic system utility commands to communicating through socket programming. With the help of socket programming you will be able to build client-server applications in no time. The "secret sauce" of this book is its curated list of topics and solutions, which fit together through a set of different pragmatic examples; each topic is covered from scratch in an easy-to-learn way. On that journey, you'll focus on practical implementations and an outline of best practices and potential pitfalls. The book also includes a bonus chapter with a list of advanced topics and directions to grow your skills. What You Will Learn Program with operating systems using the latest version of C Work with Linux Carry out multithreading with C Examine the POSIX standards Work with files, directories, processes, and signals Explore IPC and how to work with it Who This Book Is For Programmers who have an exposure to C programming and want to learn system programming. This book will help them to learn about core concepts of operating systems with the help of C programming. .

Rust in Action is a hands-on guide to systems programming with Rust. Written for inquisitive programmers, it presents real-world use cases that go far beyond syntax and structure. Summary Rust in Action introduces the Rust programming language by exploring numerous systems programming concepts and techniques. You'll be learning Rust by delving into how computers work under the hood. You'll find yourself playing with persistent storage, memory, networking and even tinkering with CPU instructions. The book takes you through using Rust to extend other applications and teaches you tricks to write blindingly fast code. You'll also discover parallel and concurrent programming. Filled to the brim with real-life use cases and scenarios, you'll go beyond the Rust syntax and see what Rust has to offer in real-world use cases. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the technology Rust is the perfect language for systems programming. It delivers the low-level power of C along with rock-solid safety features that let you code fearlessly. Ideal for applications requiring concurrency, Rust programs are compact, readable, and blazingly fast. Best of all, Rust's famously smart compiler helps you avoid even subtle coding errors. About the book Rust in Action is a hands-on guide to systems programming with Rust. Written for inquisitive programmers, it presents real-world use cases that go far beyond syntax and structure. You'll explore Rust implementations for file manipulation, networking, and kernel-level programming and discover awesome techniques for parallelism and concurrency. Along the way, you'll master Rust's unique borrow checker model for memory management without a garbage collector. What's inside Elementary to advanced Rust programming Practical examples from systems programming Command-line, graphical and networked applications About the reader For intermediate programmers. No previous experience with Rust required. About the author Tim McNamara uses Rust to build data processing pipelines and generative art. He is an expert in natural language processing and data engineering. Table of Contents 1 Introducing Rust PART 1 RUST LANGUAGE DISTINCTIVES 2 Language foundations 3 Compound data types 4 Lifetimes, ownership, and borrowing PART 2 DEMYSTIFYING SYSTEMS PROGRAMMING 5 Data in depth 6 Memory 7 Files and storage 8 Networking 9 Time and timekeeping 10 Processes, threads, and containers 11 Kernel 12 Signals, interrupts, and exceptions

Many systems today use the C programming language as it is available for most computers This book looks at how to produce C programs to execute on a PC or a MAC computer. It also looks at the Arduino UNO micro controller and describes how to write C programs using the Arduino "wired" C functions as well as using standard ANSI C with direct access to the micro controller registers of the Arduino UNO. This can lead to improved efficiency of the programs. Most of the Hardware available in the Arduino micro controller is described, and programs provided showing how to control and use them. There is a chapter on how to create your own programs and also how to change a program created to execute on the Arduino so that it can run on a different micro controller, such as the Microchip PIC. This allows the Arduino to be used as a rapid prototype system. The book also contains many working program examples with additional workshop exercises for the reader to study.

Advanced Test in C and Embedded System Programming

System Programming with C and Unix

Windows System Programming

An Introduction to Systems Programming

Hands-On Network Programming with C

Unix System Programming Using C++

A problem-solution-based guide to help you overcome hurdles effectively while working with kernel APIs, filesystems, networks, threads, and process communications **Key Features**Learn to apply the latest C++ features (from C++11, 14, 17, and 20) to facilitate systems programmingCreate robust and concurrent systems that make the most of the available hardware resourcesDelve into C++ inbuilt libraries and frameworks to design robust systems as per your business needsBook Description C++ is the preferred language for system programming due to its efficient low-level computation, data abstraction, and object-oriented features. System programming is about designing and writing computer programs that interact closely with the underlying operating system and allow computer hardware to interface with the programmer and the user. The C++ System Programming Cookbook will serve as a reference for developers who want to have ready-to-use solutions for the essential aspects of system programming using the latest C++ standards wherever possible. This C++ book starts out by giving you an overview of system programming and refreshing your C++ knowledge. Moving ahead, you will learn how to deal with threads and processes, before going on to discover recipes for how to manage memory. The concluding chapters will then help you understand how processes communicate and how to interact with the console (console I/O). Finally, you will learn how to deal with time interfaces, signals, and CPU scheduling. By the end of the book, you will become adept at developing robust systems

applications using C++. What you will learnGet up to speed with the fundamentals including makefile, man pages, compilation, and linking and debuggingUnderstand how to deal with time interfaces, signals, and CPU schedulingDevelop your knowledge of memory managementUse processes and threads for advanced synchronizations (mutexes and condition variables)Understand interprocess communications (IPC): pipes, FIFOs, message queues, shared memory, and TCP and UDPDiscover how to interact with the console (console I/O)Who this book is for This book is for C++ developers who want to gain practical knowledge of systems programming. Though no experience of Linux system programming is assumed, intermediate knowledge of C++ is necessary.

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.

Systems Programming: Designing and Developing Distributed Applications explains how the development of distributed applications depends on a foundational understanding of the relationship among operating systems, networking, distributed systems, and programming. Uniquely organized around four viewpoints (process, communication, resource, and architecture), the fundamental and essential characteristics of distributed systems are explored in ways which cut across the various traditional subject area boundaries. The structures, configurations and behaviours of distributed systems are all examined, allowing readers to explore concepts from different perspectives, and to understand systems in depth, both from the component level and holistically. Explains key ideas from the ground up, in a self-contained style, with material carefully sequenced to make it easy to absorb and follow. Features a detailed case study that is designed to serve as a common point of reference and to provide continuity across the different technical chapters. Includes a 'putting it all together' chapter that looks at interesting distributed systems applications across their entire life-cycle from requirements analysis and design specifications to fully working applications with full source code. Ancillary materials include problems and solutions, programming exercises, simulation experiments, and a wide range of fully working sample applications with complete source code developed in C++, C# and Java. Special editions of the author's established 'workbenches' teaching and learning tools suite are included. These tools have been specifically designed to facilitate practical experimentation and simulation of complex and dynamic aspects of systems.

Provides the nitty gritty details on how UNIX interacts with applications. Includes many extended examples on topics ranging from string manipulation to network programming

Designing and Developing Distributed Applications

Building a Second Brain

Communication, Concurrency, and Threads

Using C and Pep/9 Assembly

UNIX Systems Programming

C++ System Programming Cookbook

A hands-on guide to making system programming with C++ easy Key FeaturesWrite system-level code leveraging C++17Learn the internals of the Linux Application Binary Interface (ABI) and apply it to system programmingExplore C++ concurrency to take advantage of server-level constructsBook Description C++ is a general-purpose programming language with a bias toward system programming as it provides ready access to hardware-level resources, efficient compilation, and a versatile approach to higher-level abstractions. This book will help you understand the benefits of system programming with C++17. You will gain a firm understanding of various C, C++, and POSIX standards, as well as their respective system types for both C++ and POSIX. After a brief refresher on C++, Resource Acquisition Is Initialization (RAII), and the new C++ Guideline Support Library (GSL), you will learn to program Linux and Unix systems along with process management. As you progress through the chapters, you will become acquainted with C++'s support for IO. You will then study various memory management methods, including a chapter on allocators and how they benefit system programming. You will also explore how to program file input and output and learn about POSIX sockets. This book will help you get to grips with safely setting up a UDP and TCP server/client. Finally, you will be guided through Unix time interfaces, multithreading, and error handling with C++ exceptions. By the end of this book, you will be comfortable with using C++ to program high-quality systems. What you will learnUnderstand the benefits of using C++ for system programmingProgram Linux/Unix systems using C++Discover the advantages of Resource Acquisition Is Initialization (RAII)Program both console and file input and outputUncover the POSIX socket APIs and understand how to program themExplore advanced system programming topics, such as C++ allocatorsUse POSIX and C++ threads to program concurrent systemsGrasp how C++ can be used to create performant system applicationsWho this book is for If you are a fresh developer with intermediate knowledge of C++ but little or no knowledge of Unix and Linux system programming, this book will help you learn system programming with C++ in a practical way.

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.

A comprehensive guide to programming with network sockets, implementing Internet protocols, designing IoT devices, and much more with C Key FeaturesLeverage your C or C++ programming skills to build powerful network applicationsGet to grips with a variety of network protocols that allow you to load web pages, send emails, and do much moreWrite portable network code for operating systems such as Windows, Linux, and macOSBook Description Network programming, a challenging topic in C, is made easy to understand with a careful exposition of socket programming APIs. This book gets you started with modern network programming in C and the right use of relevant operating system APIs. This book covers core concepts, such as hostname resolution with DNS, that are crucial to the functioning of the modern web. You'll delve into the fundamental network protocols, TCP and UDP. Essential techniques for networking paradigms such as client-server and peer-to-peer models are explained with the help of practical examples. You'll also study HTTP and HTTPS (the protocols responsible for web pages) from both the client and server perspective. To keep up with current trends, you'll apply the concepts covered in this book to gain insights into web programming for IoT. You'll even get to grips with network monitoring and implementing security best practices. By the end of this book, you'll have experience of working with client-server applications, and be able to implement new network programs in C. The code in this book is compatible with the older C99 version as well as the latest C18 and C++17 standards. Special consideration is given to writing robust, reliable, and secure code that is portable across operating systems, including Winsock sockets for Windows and POSIX sockets for Linux and macOS. What you will learnUncover cross-platform socket programming APIsImplement techniques for supporting IPv4 and IPv6Understand how TCP and UDP connections work over IPDiscover how hostname resolution and DNS workInterface with web APIs using HTTP and HTTPSAcquire hands-on experience with Simple Mail Transfer Protocol (SMTP)Apply network programming to the Internet of Things (IoT)Who this book is for If you're a developer or a system administrator who wants to enter the world of network programming, this book is for you. Basic knowledge of C programming is assumed.

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.

Programming Embedded Systems in C and C++

Talking Directly to the Kernel and C Library

Using C on the UNIX System

Practical System Programming for Rust Developers

Linux System Programming

Hands-On System Programming with Linux

For intermediate to experienced C programmers who want to become UNIX system programmers. Explains system calls and special library routines available on the system. Annotation copyrighted by Book News, Inc., Portland, OR 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 systemsoftware and advanced programming skills, allowing readers to interface with operatingssystem 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 book doesn't assume any programming background. It begins with the basics and steadily builds the pace so that the reader finds it easy to handle advanced topics towards the end of the book. Each chapter contains:--Lucid explanation of the concept -Well thought-out, fully working programming examples -End-of-chapter exercises that would help you practise the skills learned in the chapter.CONTENTSFundamentals of ComputersProgramming BasicsDigital ComputersProblem Solving ApproachesBasic OperationsAlgorithmsFunctional ComponentsFlowchartsNumbering SystemsTypes of LanguagesBinary ArithmeticAssembler, Compiler, Linker, LoaderFundamentals of C ProgrammingBuilding Blocks of C ProgrammingStructure of a C ProgramDecision Control InstructionWriting & Executing ProgramsLoop Control InstructionStandard I/O OperationsCase Control InstructionFundamental Data TypesBreak & Continue KeywordsStorage ClassesFunctionsTypes of OperatorsParameter PassingTypes of ExpressionsRecursive FunctionsArrays & Other Data TypesPointers and Their UsageArray Notation & representationIntroduction to PointersManipulating Array ElementsTypes of PointersMulti-dimensional ArraysFile PointersStructuresFile OperationsUnionsCommand-line ArgumentsEnumsPreprocessor Directives

The Definitive Guide to Windows API Programming. Fully Updated for Windows 7, Windows Server 2008, and Windows Vista Windows System Programming, Fourth Edition, now contains extensive new coverage of 64-bit programming, parallelism, multicore systems, and many other crucial topics. Johnson Hart's robust code examples have been updated and streamlined throughout. They have been debugged and tested in both 32-bit and 64-bit versions, on single and multiprocessor systems, and under Windows 7, Vista, Server 2008, and Windows XP. To clarify program operation, sample programs are now illustrated with dozens of screenshots. Hart systematically covers Windows externals at the API level, presenting practical coverage of all the services Windows programmers need, and emphasizing how Windows functions actually behave and interact in real-world applications. Hart begins with features used in single-process applications and gradually progresses to more sophisticated functions and multithreaded environments. Topics covered include file systems, memory management, exceptions, processes, threads, synchronization, interprocess communication, Windows services, and security. New coverage in this edition includes Leveraging parallelism and maximizing performance in multicore systems Promoting source code portability and application interoperability across Windows, Linux, and UNIX Using 64-bit address spaces and ensuring 64-bit/32-bit portability Improving performance and scalability using threads, thread pools, and completion ports Techniques to improve program reliability and performance in all systems Windows performance-enhancing API features available starting with Windows Vista, such as slim reader/writer locks and condition variables A companion Web site, jmhartsoftware.com, contains all sample code, Visual Studio projects, additional examples, errata, reader comments, and Windows commentary and discussion.

Practical recipes for Linux system-level programming using the latest C++ features

Go Systems Programming

The C Programming Language

UNIX Systems Programming for SVR4

Practical Recipes for Linux System-Level Programming Using the Latest C++ Features

Windows 10 System Programming, Part 1

System Programming with C and UnixPearson Higher Ed

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

This is the eBook of the printed book and may not include any media, website access codes, or print supplements that may come packaged with the bound book. Beginning computing students often finish the introduction to programming course without having had exposure to various system tools, without knowing how to optimize program performance and without understanding how programs interact with the larger computer system. Adam Hoover's System Programming with C and Unix introduces students to commonly used system tools (libraries, debuggers, system calls, shells and scripting languages) and then explains how to utilize these tools to optimize program development. The text also examines lower level data types with an emphasis on memory and understanding how and why different data types are used.

This Book Is Heavily Inclined Towards The Requirement Of Skilled C/Embedded System Programmer. This Book Address The Need Of Less Experienced Programmer While Augmenting The Knowledge Of More Experienced Programmer. It Is Designed For All Those Aspiring For A Career In It Focusing On The C And Embedded System Programming. This Is A Unique Book To Help Prepare And Appear For The Various Screening Tests And Campus Interviews.

Practical System Programming with C

Programming Embedded Systems

Become a proficient Linux system programmer using expert recipes and techniques

Systems Programming in Unix/Linux

With C and GNU Development Tools

Pragmatic Example Applications in Linux and Unix-based Operating Systems

A problem-solution-based guide to help you overcome hurdles effectively while working with kernel APIs, filesystems, networks, threads, and process communications Key Features Learn to apply the latest C++ features (from C++11, 14, 17, and 20) to facilitate systems programming Create robust and concurrent systems that make the most of the available hardware resources Delve into C++ inbuilt libraries and frameworks to design robust systems as per your business needs Book Description C++ is the preferred language for system programming due to its efficient low-level computation, data abstraction, and object-oriented features. System programming is about designing and writing computer programs that interact closely with the underlying operating system and allow computer hardware to interface with the programmer and the user. The C++ System Programming Cookbook will serve as a reference for developers who want to have ready-to-use solutions for the essential aspects of system programming using the latest C++ standards wherever possible. This C++ book starts out by giving you an overview of system programming and refreshing your C++ knowledge. Moving ahead, you will learn how to deal with threads and processes, before going on to discover recipes for how to manage memory. The concluding chapters will then help you understand how processes communicate and how to interact with the console (console I/O). Finally, you will learn how to deal with time interfaces, signals, and CPU scheduling. By the end of the book, you will become adept at developing robust systems applications using C++. What you will learn Get up to speed with the fundamentals including makefile, man pages, compilation, and linking and debugging Understand how to deal with time interfaces, signals, and CPU scheduling Develop your knowledge of memory management Use processes and threads for advanced synchronizations (mutexes and condition variables) Understand interprocess communications (IPC): pipes, FIFOs, message queues, shared memory, and TCP and UDP Discover how to interact with the console (console I/O) Who this book is for This book is for C++ developers who want to gain practical knowledge of systems programming. Though no experience of Linux system programming is assumed, intermediate knowledge of C++ is necessary. When I was asked to teach a class in computer architecture and assembly language, I was excited since it allowed me to utilize the portions of my career where I worked with computer and chip manufacturers. At the same time, the task of developing the course was rather daunting especially when I was told there would be no prerequisites: students could range from those with no previous computer classes to students who have taken many computer classes. I set myself to developing a class that would introduce computers and programming to those new to computer science and at the same time teach the more experienced students systems development and programming techniques they may not have picked up in previous classes. After teaching the course a couple of times, students told me that they could benefit from a textbook that more closely resembled the class material I was presenting. While on vacation in Santa Fe, New Mexico, I woke up early one morning and decided to write down some thoughts on binary logic. During the vacation, I completed the chapter save some editing. With one down, I thought I would create at least a collection of three: binary numbers, binary logic, and binary representations. Then, I decided I would write two chapters on programming (C and assembly) that would parallel each other and could be reinforced with weekly programming projects. By then, I had covered most of the course and it was a matter of filling in the gaps. For students who have no experience in computer programming, I think you will find this text provides a useful understanding of computers that can be a foundation for your future classes. If you are not going to pursue computer science as a career, you may find yourself using some of what is taught here in any career that uses computers. For students who have some experience with computer programming, some of the programming topics discussed in this text may be a bit redundant; however, your previous programming classes likely focused on application development. The use of the C programming language and assembly language and this text are geared more toward systems programming. Unlike application programming, systems programming is more tied to the underlying hardware architecture than application programming. Because of this, systems programming and assembly language help reinforce an understanding of computer architecture.

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.

A Linux and UNIX System Programming Handbook

Hands-On System Programming with C++

Practical Systems Programming with C

A Proven Method to Organize Your Digital Life and Unlock Your Creative Potential

System Software