

## Rust Programming Language The Manga Guide

What if William Shakespeare were asked to generate the Fibonacci series or Jane Austen had to write a factorial program? In *If Hemingway Wrote JavaScript*, author Angus Croll imagines short JavaScript programs as written by famous wordsmiths. The result is a peculiar and charming combination of prose, poetry, and programming. The best authors are those who obsess about language—and the same goes for JavaScript developers. To master either craft, you must experiment with language to develop your own style, your own idioms, and your own expressions. To that end, *If Hemingway Wrote JavaScript* playfully bridges the worlds of programming and literature for the literary geek in all of us. Featuring original artwork by Miran Lipovača.

*Computer Graphics from Scratch* demystifies the algorithms used in modern graphics software and guides beginners through building photorealistic 3D renders. Computer graphics programming books are often math-heavy and intimidating for newcomers. Not this one. *Computer Graphics from Scratch* takes a simpler approach by keeping the math to a minimum and focusing on only one aspect of computer graphics, 3D rendering. You'll build two complete, fully functional renderers: a raytracer, which simulates rays of light as they bounce off objects, and a rasterizer, which converts 3D models into 2D pixels. As you progress you'll learn how to create realistic reflections and shadows, and how to render a scene from any point of view. Pseudocode examples throughout make it easy to write your renderers in any language, and links to live JavaScript demos of each algorithm invite you to explore further on your own. Learn how to:

- Use perspective projection to draw 3D objects on a 2D plane
- Simulate the way rays of light interact with surfaces
- Add mirror-like reflections and cast shadows to objects
- Render a scene from any camera position using clipping planes
- Use flat, Gouraud, and Phong shading to mimic real surface lighting
- Paint texture details onto basic shapes to create realistic-looking objects

Whether you're an aspiring graphics engineer or a novice programmer curious about how graphics algorithms work, Gabriel Gambetta's simple, clear explanations will quickly put computer graphics concepts and rendering techniques within your reach. All you need is basic coding knowledge and high school math. *Computer Graphics from Scratch* will cover the rest.

A project-filled introduction to coding that shows kids how to build programs by making cool games. Scratch, the colorful drag-and-drop programming language, is used by millions of first-time learners worldwide. Scratch 3 features an updated interface, new programming blocks, and the ability to run on tablets and smartphones, so you can learn how to code on the go. In *Scratch 3 Programming Playground*, you'll learn to code by making cool games. Get ready to destroy asteroids, shoot hoops, and slice and dice fruit! Each game includes easy-to-follow instructions with full-color images, review questions, and creative coding challenges to make the game your own. Want to add more levels or a cheat code? No problem, just write some code. You'll learn to make games like:

- Maze Runner: escape the maze!
- Snaaaaaake: gobble apples and avoid your own tail
- Asteroid Breaker: smash space rocks
- Fruit Slicer: a Fruit Ninja clone
- Brick Breaker: a remake of Breakout, the brick-breaking classic
- Platformer: a game inspired by Super Mario Bros

Learning how to program shouldn't be dry and dreary. With *Scratch 3 Programming Playground*, you'll make a game of it! Covers: Scratch 3

Key concepts and best practices for new software engineers — stuff critical to your workplace success that you weren't taught in school. For new software engineers, knowing how to program is only half the battle. You'll quickly find that many of the skills and processes key to your success are not taught in any school or bootcamp. *The Missing README* fills in that gap—a distillation of workplace lessons, best practices, and engineering fundamentals that the authors have taught rookie developers at top companies for more than a decade. Early chapters explain what to expect when you begin your career at a company. The book's middle section expands your technical education, teaching you how to work with existing codebases, address and prevent technical debt, write production-grade software, manage dependencies, test effectively, do code reviews, safely deploy software, design evolvable architectures, and handle incidents when you're on-call. Additional chapters cover planning and interpersonal skills such as Agile planning, working effectively with your manager, and growing to senior levels and beyond. You'll learn:

- How to use the legacy code change algorithm, and leave code cleaner than you found it
- How to write operable code with logging, metrics, configuration, and defensive programming
- How to write deterministic tests, submit code reviews, and give feedback on other people's code
- The technical design process, including experiments, problem definition, documentation, and collaboration
- What to do when you are on-call, and how to navigate production incidents
- Architectural techniques that make code change easier
- Agile development practices like sprint planning, stand-ups, and retrospectives

This is the book your tech lead wishes every new engineer would read before they start. By the end, you'll know what it takes to transition into the workplace—from CS classes or bootcamps to professional software engineering.

Website security made easy. This book covers the most common ways websites get hacked and how web developers can defend themselves. The world has changed. Today, every time you make a site live, you're opening it up to attack. A first-time developer can easily be discouraged by the difficulties involved with properly securing a website. But have hope: an army of security researchers is out there discovering, documenting, and fixing security flaws. Thankfully, the tools you'll need to secure your site are freely available and generally easy to use. *Web Security for Developers* will teach you how your websites are vulnerable to attack and how to protect them. Each chapter breaks down a major security vulnerability and explores a real-world attack, coupled with plenty of code to show you both the vulnerability and the fix. You'll learn how to:

- Protect against SQL injection attacks, malicious JavaScript, and cross-site request forgery
- Add authentication and shape access control to protect accounts
- Lock down user accounts to prevent attacks that rely on guessing passwords, stealing sessions, or escalating privileges
- Implement encryption
- Manage vulnerabilities in legacy code
- Prevent information leaks that disclose vulnerabilities
- Mitigate advanced attacks like malvertising and denial-of-service

As you get stronger at identifying and fixing vulnerabilities, you'll learn to deploy disciplined, secure code and become a better programmer along the way.

*The Go Programming Language* is the authoritative resource for any programmer who wants to learn Go. It shows how to write clear and idiomatic Go to solve real-world problems. The book does not assume prior knowledge of Go nor experience with any specific language, so you'll find it accessible whether you're most comfortable with JavaScript, Ruby, Python, Java, or C++. The first chapter is a tutorial on the basic concepts of Go, introduced through programs for file I/O and text processing, simple graphics, and web clients and servers. Early chapters cover the structural elements of Go programs: syntax, control flow, data types, and the organization of a program into packages, files, and functions. The examples illustrate many packages from the standard library and show how to create new ones of your own. Later chapters explain the package mechanism in more detail, and how to build,

test, and maintain projects using the go tool. The chapters on methods and interfaces introduce Go's unconventional approach to object-oriented programming, in which methods can be declared on any type and interfaces are implicitly satisfied. They explain the key principles of encapsulation, composition, and substitutability using realistic examples. Two chapters on concurrency present in-depth approaches to this increasingly important topic. The first, which covers the basic mechanisms of goroutines and channels, illustrates the style known as communicating sequential processes for which Go is renowned. The second covers more traditional aspects of concurrency with shared variables. These chapters provide a solid foundation for programmers encountering concurrency for the first time. The final two chapters explore lower-level features of Go. One covers the art of metaprogramming using reflection. The other shows how to use the unsafe package to step outside the type system for special situations, and how to use the cgo tool to create Go bindings for C libraries. The book features hundreds of interesting and practical examples of well-written Go code that cover the whole language, its most important packages, and a wide range of applications. Each chapter has exercises to test your understanding and explore extensions and alternatives. Source code is freely available for download from <http://gopl.io/> and may be conveniently fetched, built, and installed using the go get command.

Systems programming provides the foundation for the world's computation. Writing performance-sensitive code requires a programming language that puts programmers in control of how memory, processor time, and other system resources are used. The Rust systems programming language combines that control with a modern type system that catches broad classes of common mistakes, from memory management errors to data races between threads. With this practical guide, experienced systems programmers will learn how to successfully bridge the gap between performance and safety using Rust. Jim Blandy, Jason Orendorff, and Leonora Tindall demonstrate how Rust's features put programmers in control over memory consumption and processor use by combining predictable performance with memory safety and trustworthy concurrency. You'll learn: Rust's fundamental data types and the core concepts of ownership and borrowing How to write flexible, efficient code with traits and generics How to write fast, multithreaded code without data races Rust's key power tools: closures, iterators, and asynchronous programming Collections, strings and text, input and output, macros, unsafe code, and foreign function interfaces This revised, updated edition covers the Rust 2021 Edition.

Like the best-selling Black Hat Python, Black Hat Go explores the darker side of the popular Go programming language. This collection of short scripts will help you test your systems, build and automate tools to fit your needs, and improve your offensive security skillset. Black Hat Go explores the darker side of Go, the popular programming language revered by hackers for its simplicity, efficiency, and reliability. It provides an arsenal of practical tactics from the perspective of security practitioners and hackers to help you test your systems, build and automate tools to fit your needs, and improve your offensive security skillset, all using the power of Go. You'll begin your journey with a basic overview of Go's syntax and philosophy and then start to explore examples that you can leverage for tool development, including common network protocols like HTTP, DNS, and SMB. You'll then dig into various tactics and problems that penetration testers encounter, addressing things like data pilfering, packet sniffing, and exploit development. You'll create dynamic, pluggable tools before diving into cryptography, attacking Microsoft Windows, and implementing steganography. You'll learn how to:

- Make performant tools that can be used for your own security projects
- Create usable tools that interact with remote APIs
- Scrape arbitrary HTML data
- Use Go's standard package, net/http, for building HTTP servers
- Write your own DNS server and proxy
- Use DNS tunneling to establish a C2 channel out of a restrictive network
- Create a vulnerability fuzzer to discover an application's security weaknesses
- Use plug-ins and extensions to future-proof products
- Build an RC2 symmetric-key brute-forcer
- Implant data within a Portable Network Graphics (PNG) image.

Are you ready to add to your arsenal of security tools? Then let's Go!

A study of the relationship between platform and creative expression in the Atari VCS. The Atari Video Computer System dominated the home video game market so completely that "Atari" became the generic term for a video game console. The Atari VCS was affordable and offered the flexibility of changeable cartridges. Nearly a thousand of these were created, the most significant of which established new techniques, mechanics, and even entire genres. This book offers a detailed and accessible study of this influential video game console from both computational and cultural perspectives. Studies of digital media have rarely investigated platforms—the systems underlying computing. This book (the first in a series of Platform Studies) does so, developing a critical approach that examines the relationship between platforms and creative expression. Nick Montfort and Ian Bogost discuss the Atari VCS itself and examine in detail six game cartridges: Combat, Adventure, Pac-Man, Yars' Revenge, Pitfall!, and Star Wars: The Empire Strikes Back. They describe the technical constraints and affordances of the system and track developments in programming, gameplay, interface, and aesthetics. Adventure, for example, was the first game to represent a virtual space larger than the screen (anticipating the boundless virtual spaces of such later games as World of Warcraft and Grand Theft Auto), by allowing the player to walk off one side into another space; and Star Wars: The Empire Strikes Back was an early instance of interaction between media properties and video games. Montfort and Bogost show that the Atari VCS—often considered merely a retro fetish object—is an essential part of the history of video games.

The Rust Programming Language is the official, definitive guide to Rust, a hugely popular, community-supported programming language. The reader will learn all about Rust's ownership rules, which lie at the heart of Rust's reliability and crash-resistant compiling. The Rust Programming Language covers everything from basic concepts like variable bindings, control flow, functions, and error handling, to more advanced topics, such as crates, generics, concurrency, and the nitty gritty of Rust's type system.

A primer on the underlying technologies that allow computer programs to work. Covers topics like computer hardware, combinatorial logic, sequential logic, computer architecture, computer anatomy, and Input/Output. Many coders are unfamiliar with the underlying technologies that make their programs run. But why should you care when your code appears to work? Because you want it to run well and not be riddled with hard-to-find bugs. You don't want to be in the news because your code had a security problem. Lots of technical detail is available online but it's not organized or collected into a convenient place. In The Secret Life of Programs, veteran engineer Jonathan E. Steinhart explores—in depth—the foundational concepts that underlie the machine. Subjects like computer hardware, how software behaves on hardware, as well as how people have solved problems using technology over time. You'll learn:

- How the real world is converted into a form that computers understand, like bits, logic, numbers, text, and colors
- The fundamental building blocks that make up a computer including logic gates, adders, decoders, registers, and memory
- Why designing programs to match computer hardware, especially memory, improves performance
- How programs are converted into machine language that computers understand
- How software building blocks are combined to create programs like web browsers
- Clever tricks for making programs more efficient, like loop invariance, strength reduction, and recursive subdivision
- The fundamentals of computer security and machine intelligence
- Project design, documentation, scheduling, portability, maintenance, and other practical programming realities.

Learn what really happens when your code runs on the machine and you'll learn to craft better, more efficient code.

Build projects on exciting topics like game development, virtual reality, web assembly, emulators, GUI, and Linux/kernel development. By the

end of the book, you will know how to choose the right framework or library for your needs.

This book constitutes the refereed post-conference proceedings of 10 workshops held at the 35th International ISC High Performance 2020 Conference, in Frankfurt, Germany, in June 2020: First Workshop on Compiler-assisted Correctness Checking and Performance Optimization for HPC (C3PO); First International Workshop on the Application of Machine Learning Techniques to Computational Fluid Dynamics Simulations and Analysis (CFDML); HPC I/O in the Data Center Workshop (HPC-IODC); First Workshop "Machine Learning on HPC Systems" (MLHPCS); First International Workshop on Monitoring and Data Analytics (MODA); 15th Workshop on Virtualization in High-Performance Cloud Computing (VHPC). The 25 full papers included in this volume were carefully reviewed and selected. They cover all aspects of research, development, and application of large-scale, high performance experimental and commercial systems. Topics include high-performance computing (HPC), computer architecture and hardware, programming models, system software, performance analysis and modeling, compiler analysis and optimization techniques, software sustainability, scientific applications, deep learning.

A thorough analysis of contemporary digital media practices, showing how people increasingly not only consume but also produce and even design media. With many new forms of digital media—including such popular social media as Facebook, Twitter, and Flickr—the people formerly known as the audience no longer only consume but also produce and even design media. Jonas Löwgren and Bo Reimer term this phenomenon collaborative media, and in this book they investigate the qualities and characteristics of these forms of media in terms of what they enable people to do. They do so through an interdisciplinary research approach that combines the social sciences and humanities traditions of empirical and theoretical work with practice-based, design-oriented interventions. Löwgren and Reimer offer analysis and a series of illuminating case studies—examples of projects in collaborative media that range from small multidisciplinary research experiments to commercial projects used by millions of people. Löwgren and Reimer discuss the case studies at three levels of analysis: society and the role of collaborative media in societal change; institutions and the relationship of collaborative media with established media structures; and tribes, the nurturing of small communities within a large technical infrastructure. They conclude by advocating an interventionist turn within social analysis and media design.

The Ruby programming language is perfect for beginners: easy to learn, powerful, and fun to use! But wouldn't it be more fun if you were learning with the help of some wizards and dragons? Ruby Wizardry is a playful, illustrated tale that will teach you how to program in Ruby by taking you on a fantastical journey. As you follow the adventures of young heroes Ruben and Scarlet, you'll learn real programming skills, like how to: –Use fundamental concepts like variables, symbols, arrays, and strings –Work with Ruby hashes to create a programmable breakfast menu –Control program flow with loops and conditionals to help the Royal Plumber –Test your wild and crazy ideas in IRB and save your programs as scripts –Create a class of mini-wizards, each with their own superpower! –Organize and reuse your code with methods and lists –Write your own amazing interactive stories using Ruby Along the way, you'll meet colorful characters from around the kingdom, like the hacker Queen, the Off-White Knight, and Wherefore the minstrel. Ruby Wizardry will have you (or your little wizard) hooked on programming in no time. For ages 10+ (and their parents!)

Ayumi is a world-class shogi (Japanese chess) player who can't be beaten—that is, until she loses to a powerful computer called the Shooting Star. Ayumi vows to find out everything she can about her new nemesis. Lucky for her, Yuu Kano, the genius programmer behind the Shooting Star, is willing to teach her all about the inner workings of the microprocessor—the "brain" inside all computers, phones, and gadgets. Follow along with Ayumi in The Manga Guide to Microprocessors and you'll learn about: -How the CPU processes information and makes decision -How computers perform arithmetic operations and store information -logic gates and how they're used in integrated circuits -the Key components of modern computers, including registers, GPUs, and RAM -Assembly language and how it differs from high-level programming languages Whether you're a computer science student or just want to understand the power of microprocessors, you'll find what you need to know in The Manga Guide to Microprocessors.

Rust is an exciting new programming language combining the power of C with memory safety, fearless concurrency, and productivity boosters - and what better way to learn than by making games. Each chapter in this book presents hands-on, practical projects ranging from "Hello, World" to building a full dungeon crawler game. With this book, you'll learn game development skills applicable to other engines, including Unity and Unreal. Rust is an exciting programming language combining the power of C with memory safety, fearless concurrency, and productivity boosters. With Rust, you have a shiny new playground where your game ideas can flourish. Each chapter in this book presents hands-on, practical projects that take you on a journey from "Hello, World" to building a full dungeon crawler game. Start by setting up Rust and getting comfortable with your development environment. Learn the language basics with practical examples as you make your own version of Flappy Bird. Discover what it takes to randomly generate dungeons and populate them with monsters as you build a complete dungeon crawl game. Run game systems concurrently for high-performance and fast game-play, while retaining the ability to debug your program. Unleash your creativity with magical items, tougher monsters, and intricate dungeon design. Add layered graphics and polish your game with style. What You Need: A computer running Windows 10, Linux, or Mac OS X. A text editor, such as Visual Studio Code. A video card and drivers capable of running OpenGL 3.2.

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

YOKO'S TRUTH Alita and Erica escape Gillatin unscathed, but end up in the hands of Duke Marquis, who intends to install Alita as the new Lady of Cydonia for his own nefarious purposes. But a hitch in the Ritual of Succession unfolds into a shocking revelation as the secret origin of Yoko, the girl who will one day become Alita, is revealed at last!

Racket is a descendant of Lisp, a programming language renowned for its elegance, power, and challenging learning curve. But while Racket retains the functional goodness of Lisp, it was designed with beginning programmers in mind. Realm of Racket is your introduction to the Racket language. In Realm of Racket, you'll learn to program by creating increasingly complex games. Your journey begins with the Guess My Number game and coverage of some basic Racket etiquette. Next you'll dig into syntax and semantics, lists, structures, and conditionals, and learn to work with recursion and the GUI as you build the Robot Snake game. After that it's on to lambda and mutant structs (and an Orc Battle), and fancy loops and the Dice of Doom. Finally, you'll explore laziness, AI, distributed games, and the Hungry Henry game. As you progress through the games, chapter checkpoints and challenges help reinforce what you've learned. Offbeat comics keep things fun along the way. As you travel through the Racket realm, you'll: –Master the quirks of Racket's syntax and semantics –Learn to write concise and elegant functional programs –Create a graphical user interface using the 2htdp/image library –Create a server to handle true multiplayer games Realm of Racket is a lighthearted guide to some serious programming. Read it to see why Racketeers have so much fun!

In the near future, a machinist, a monk, and a mesmerizer are looking for the Source of the universe, but among the ancient scriptures, they only find the devil ... Few humans in the future find the secrets of what words and sounds can do, for the Apocalypse and war have led them to evolve in mind and physique. Two of them end World War III with just a four-minute speech. Some say they did mass hypnosis. Others say it was their voice and will. Fifty years later in South Asia, KUSHA, a twenty-three-year-old machine-geek with social awkwardness and amnesia, tries to get the Devil's Book with secrets of voice. But her idol of voice and everyone's beloved war heroes, YUAN and RUEM, are also after it for power. In a society that worships the evolved High Grades with voice, how you speak and which words you talk with is important. As someone who finds all solutions in books, Kusha thinks the secrets in the three-foot-long ancient book will teach her to speak mesmerizingly. She believes it will help her evolve. So, she decides to attend the auction where the book will be sold. But there's a problem; powerful High Grades want the book too. They want to code fate, rewriting the material world. They are undead, immortal mesmerizers. And being a philosopher, Kusha admires their ethics while she must fight them—the fight that starts with the book. The fight creates a chain reaction leading her to an epic journey. About the Series: THE HIGH AUCTION: WISDOM REVOLUTION is a book of genre-bending Adult Sci-fi-Fantasy series "The Machinist, The Monk, And The Mesmerizer Chronicles." Here, you'll find Metaphysical magic built around philosophy and spirituality. You'll find ancient mysteries, archaeological adventures, and, most importantly, you will meet some characters. Its inspiration was from myths/truth (read the word you prefer) of all religions where humans evolved to Sufis, Rishis, Sannashis, Monks, and sometimes, Gods. Example: Shiva Purana in Hindu epics where Sati evolved to Goddess Parvati and splits into Kali to destroy the evil is a prominent example. You'll find diverse cultures, sexuality, and faith in this series. Enjoy. What the Critiques and Reviewers are saying about it: "GEM PROSE ... YOU WILL FINISH THIS BOOK ONCE YOU START ..." "EXCEPTIONALLY THOUGHT-PROVOKING ..." "RELATABLE ... YOU KEEP NODDING AT EVERYTHING ..."

This is not your typical programming book! Jump right in with interesting, useful programs, some of which are drawn from classic computer science problems as a way of talking about the programming constructs in the language rather than explaining everything in a dry, theoretical manner that doesn't translate well to implementation. Rust programming has been the "most loved programming language" in the Stack Overflow Developer Survey every year since 2016! Learn why programmers are using Rust due to its performance and efficiency, without the errors and crashes that a programmer would find in common languages such as C and C++. Built around solving real problems, this book will help introduce you to computer science problems that can be built upon to create solutions for other problems. LEARN BY DOING: This book will focus on a practical approach to learning Rust. You will learn all of the language fundamentals through the use of programming examples that do interesting things! All of the programs covered will be based on a computer science problem or other interesting problems that can be used as a foundation for demonstrating language syntax, data types and structures, and other features or techniques for developing programs.

My Kids Just Gave Me a Computer, What Do I Do Now? Computers for Seniors is a step-by-step, full-color guide that will take you all the way from pressing the "On" button on your new computer to being a confident user who can send email to family and friends, shop online safely, read the latest news, watch funny YouTube videos, share cute pictures of your grandkids, check the weather forecast, and much more. You'll learn to: -Plug in, set up, and turn on your computer -Print and share photos of your grandkids, vacations, pets, friends, and special life events -Install helpful tools like a calendar, money manager, and weather tracker -Search the internet for news, recipes, gardening tips, sports updates, and anything else that interests you -Watch entertaining YouTube videos or educational lectures and make video calls to anywhere in the world -Find and listen to new music (or your favorite classics) and read electronic books -Email your friends and family -Stay safe online and keep your private information secure Computers for Seniors will show you how to

get what you really want from your PC, with the help of full-color illustrations, friendly instructions, and a touch of humor. Each lesson has small exercises to test your skills and help you practice, to make sure you feel comfortable with what you've learned before you move on. It's never too late to have fun and get more out of your PC—Computers for Seniors will ease you into the computer generation by guiding you every step of the way.

When it comes to choosing, using, and maintaining a database, understanding its internals is essential. But with so many distributed databases and tools available today, it's often difficult to understand what each one offers and how they differ. With this practical guide, Alex Petrov guides developers through the concepts behind modern database and storage engine internals. Throughout the book, you'll explore relevant material gleaned from numerous books, papers, blog posts, and the source code of several open source databases. These resources are listed at the end of parts one and two. You'll discover that the most significant distinctions among many modern databases reside in subsystems that determine how storage is organized and how data is distributed. This book examines: Storage engines: Explore storage classification and taxonomy, and dive into B-Tree-based and immutable Log Structured storage engines, with differences and use-cases for each Storage building blocks: Learn how database files are organized to build efficient storage, using auxiliary data structures such as Page Cache, Buffer Pool and Write-Ahead Log Distributed systems: Learn step-by-step how nodes and processes connect and build complex communication patterns Database clusters: Which consistency models are commonly used by modern databases and how distributed storage systems achieve consistency

A fast-paced, thorough introduction to modern C++ written for experienced programmers. After reading C++ Crash Course, you'll be proficient in the core language concepts, the C++ Standard Library, and the Boost Libraries. C++ is one of the most widely used languages for real-world software. In the hands of a knowledgeable programmer, C++ can produce small, efficient, and readable code that any programmer would be proud of. Designed for intermediate to advanced programmers, C++ Crash Course cuts through the weeds to get you straight to the core of C++17, the most modern revision of the ISO standard. Part 1 covers the core of the C++ language, where you'll learn about everything from types and functions, to the object life cycle and expressions. Part 2 introduces you to the C++ Standard Library and Boost Libraries, where you'll learn about all of the high-quality, fully-featured facilities available to you. You'll cover special utility classes, data structures, and algorithms, and learn how to manipulate file systems and build high-performance programs that communicate over networks. You'll learn all the major features of modern C++, including: • Fundamental types, reference types, and user-defined types • The object lifecycle including storage duration, memory management, exceptions, call stacks, and the RAII paradigm • Compile-time polymorphism with templates and run-time polymorphism with virtual classes • Advanced expressions, statements, and functions • Smart pointers, data structures, dates and times, numerics, and probability/statistics facilities • Containers, iterators, strings, and algorithms • Streams and files, concurrency, networking, and application development With well over 500 code samples and nearly 100 exercises, C++ Crash Course is sure to help you build a strong C++ foundation.

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"

"Fascinating. Doidge's book is a remarkable and hopeful portrait of the endless adaptability of the human brain."—Oliver Sacks, MD, author of *The Man Who Mistook His Wife for a Hat* What is neuroplasticity? Is it possible to change your brain? Norman Doidge's inspiring guide to the new brain science explains all of this and more An astonishing new science called neuroplasticity is overthrowing the centuries-old notion that the human brain is immutable, and proving that it is, in fact, possible to change your brain. Psychoanalyst, Norman Doidge, M.D., traveled the country to meet both the brilliant scientists championing neuroplasticity, its healing powers, and the people whose lives they've transformed—people whose mental limitations, brain damage or brain trauma were seen as unalterable. We see a woman born with half a brain that rewired itself to work as a whole, blind people who learn to see, learning disorders cured, IQs raised, aging brains rejuvenated, stroke patients learning to speak, children with cerebral palsy learning to move with more grace, depression and anxiety disorders successfully treated, and lifelong character traits changed. Using these marvelous stories to probe mysteries of the body, emotion, love, sex, culture, and education, Dr. Doidge has written an immensely moving, inspiring book that will permanently alter the way we look at our brains, human nature, and human potential.

Python is a powerful programming language that's easy to learn and fun to play with. But once you've gotten a handle on the basics, what do you do next? Python Playground is a collection of imaginative programming projects that will inspire you to use Python to make art and music, build simulations of real-world phenomena, and interact with hardware like the Arduino and Raspberry Pi. You'll learn to use common Python tools and libraries like numpy, matplotlib, and pygame to do things like: –Generate Spirograph-like patterns using parametric equations and the turtle module –Create music on your computer by simulating frequency overtones –Translate graphical images into ASCII art –Write an autostereogram program that produces 3D images hidden beneath random patterns –Make realistic animations with OpenGL shaders by exploring particle systems, transparency, and billboard techniques –Construct 3D visualizations using data from CT and MRI scans –Build a laser show that responds to music by hooking up your computer to an Arduino Programming shouldn't be a chore. Have some solid, geeky fun with Python Playground. The projects in this book are compatible with both Python 2 and 3.

Raised in a paranormal research facility, psychic Kai Kadou is assigned by his mentor, Eiji Sagimiya, to expose a guerrilla band that is using illegal psychics for personal gain—an effort during which he falls into the hands of an outsider who questionst

The fight to make Rin-chan the most popular Vocaloid has never been so much fun, as her smallest everyday activities at school, home, and work get blown up online to major celebrity events! Based on the video with over 2.5 million combined views on NicoNico and YouTube, the Rin-Chan Now! manga is written, drawn and edited by the video creators! Hatsune Miku is an international pop phenomenon! This is the Second of a Four part series on Hatsune Miku's RinChan! • Like Hatsune Miku: Mikubon, Rin-Chan Now! is made by famous Vocaloid music video creators. • The popular Rin was in Unofficial Hatsune Mix and was on the cover of Mikubon. • Four! pages in full color. • In the larger Unofficial Hatsune Mix page size.

Master professional-level coding in Rust. For developers who've mastered the basics, this book is the next step on your way to professional-level programming in Rust. It covers everything you need to build and maintain larger code bases, write powerful and flexible applications and

libraries, and confidently expand the scope and complexity of your projects. Author Jon Gjengset takes you deep into the Rust programming language, dissecting core topics like ownership, traits, concurrency, and unsafe code. You'll explore key concepts like type layout and trait coherence, delve into the inner workings of concurrent programming and asynchrony with `async/await`, and take a tour of the world of `no_std` programming. Gjengset also provides expert guidance on API design, testing strategies, and error handling, and will help develop your understanding of foreign function interfaces, object safety, procedural macros, and much more. You'll Learn:

- How to design reliable, idiomatic, and ergonomic Rust programs based on best principles
- Effective use of declarative and procedural macros, and the difference between them
- How asynchrony works in Rust – all the way from the `Pin` and `Waker` types used in manual implementations of `Futures`, to how `async/await` saves you from thinking about most of those words
- What it means for code to be unsafe, and best practices for writing and interacting with unsafe functions and traits
- How to organize and configure more complex Rust projects so that they integrate nicely with the rest of the ecosystem
- How to write Rust code that can interoperate with non-Rust libraries and systems, or run in constrained and embedded environments

Brimming with practical, pragmatic insights that you can immediately apply, *Rust for Rustaceans* helps you do more with Rust, while also teaching you its underlying mechanisms.

Vim is a fast and efficient text editor that will make you a faster and more efficient developer. It's available on almost every OS, and if you master the techniques in this book, you'll never need another text editor. In more than 120 Vim tips, you'll quickly learn the editor's core functionality and tackle your trickiest editing and writing tasks. This beloved bestseller has been revised and updated to Vim 7.4 and includes three brand-new tips and five fully revised tips. A highly configurable, cross-platform text editor, Vim is a serious tool for programmers, web developers, and sysadmins who want to raise their game. No other text editor comes close to Vim for speed and efficiency; it runs on almost every system imaginable and supports most coding and markup languages. Learn how to edit text the "Vim way": complete a series of repetitive changes with The Dot Formula using one keystroke to strike the target, followed by one keystroke to execute the change. Automate complex tasks by recording your keystrokes as a macro. Discover the "very magic" switch that makes Vim's regular expression syntax more like Perl's. Build complex patterns by iterating on your search history. Search inside multiple files, then run Vim's substitute command on the result set for a project-wide search and replace. All without installing a single plugin! Three new tips explain how to run multiple `ex` commands as a batch, autocomplete sequences of words, and operate on a complete search match. *Practical Vim, Second Edition* will show you new ways to work with Vim 7.4 more efficiently, whether you're a beginner or an intermediate Vim user. All this, without having to touch the mouse. What You Need: Vim version 7.4

*Hands-On Data Structures and Algorithms with Rust* will help you in upgrading your earlier knowledge of Rust so that you shift to a confident developer by implementing the algorithms in a practical environment. This would be an essential reference guide for end-user/reader to understand the fundamental techniques of Rust. This guide will cover ...

Rust is a new and fast programming language that provides memory safety without a garbage collector. With its low memory footprint, it allows web developers to build high-performance and secure web apps with relative ease. This book will help web developers to adopt Rust for web app development, while addressing safety and high-performance issues.

A clear and practical guide to building games in libGDX. This book is great for Indie and existing game developers, as well as those who want to get started with game development using libGDX. Java game knowledge of game development basics is recommended.

The Rust Programming Language No Starch Press

Discover the world of Rust programming through real-world examples

**Key Features** Implement various features of Rust to build blazingly fast applications  
Learn to build GUI applications using `Gtk-rs`  
Explore the multi-threading aspect of Rust to tackle problems in concurrency and in distributed environments

**Book Description** Rust is an open source, safe, concurrent, practical language created by Mozilla. It runs blazingly fast, prevents segfaults, and guarantees safety. This book gets you started with essential software development by guiding you through the different aspects of Rust programming. With this approach, you can bridge the gap between learning and implementing immediately. Beginning with an introduction to Rust, you'll learn the basic aspects such as its syntax, data types, functions, generics, control flows, and more. After this, you'll jump straight into building your first project, a Tetris game. Next you'll build a graphical music player and work with fast, reliable networking software using `Tokio`, the scalable and productive asynchronous IO Rust library. Over the course of this book, you'll explore various features of Rust Programming including its `SDL` features, event loop, File I/O, and the famous `GTK+` widget toolkit. Through these projects, you'll see how well Rust performs in terms of concurrency—including parallelism, reliability, improved performance, generics, macros, and thread safety. We'll also cover some asynchronous and reactive programming aspects of Rust. By the end of the book, you'll be comfortable building various real-world applications in Rust. What you will learn

Compile and run the Rust projects using the `Cargo-Rust` Package manager  
Use `Rust-SDL` features such as the event loop, windows, infinite loops, pattern matching, and more  
Create a graphical interface using `Gtk-rs` and `Rust-SDL`  
Incorporate concurrency mechanism and multi-threading along with thread safety and locks  
Implement the FTP protocol using an Asynchronous I/O stack with the `Tokio` library

Who this book is for This book is for software developers interested in system level and application programming who are looking for a quick entry into using Rust and understanding the core features of the Rust Programming. It's assumed that you have a basic understanding of Java, C#, Ruby, Python, or JavaScript.

The real challenge of programming isn't learning a language's syntax—it's learning to creatively solve problems so you can build something great. In this one-of-a-kind text, author V. Anton Spraul breaks down the ways that programmers solve problems and teaches you what other introductory books often ignore: how to Think Like a Programmer. Each chapter tackles a single programming concept, like classes, pointers, and recursion, and open-ended exercises throughout challenge you to apply your knowledge. You'll also learn how to:

- Split problems into discrete components to make them easier to solve
- Make the most of code reuse with functions, classes, and libraries
- Pick the perfect data structure for a particular job
- Master more advanced programming tools like recursion and dynamic memory
- Organize your thoughts and develop strategies to tackle particular types of problems

Although the book's examples are written in C++, the creative problem-solving concepts they illustrate go beyond any particular language; in fact, they often reach outside the realm of computer science. As the most skillful programmers know, writing great code is a creative art—and the first step in creating your masterpiece is learning to Think Like a Programmer.

How to confront, embrace, and learn from the unavoidable failures of creative practice; with case studies that range from winemaking to animation. Failure is an inevitable part of any creative practice. As game designers, John Sharp and Colleen Macklin have grappled with crises of creativity, false starts, and bad outcomes. Their tool for coping with the many varieties of failure: iteration, the cyclical process of conceptualizing, prototyping, testing, and evaluating. Sharp and Macklin have found that failure—often hidden, covered up, a source of embarrassment—is the secret ingredient of iterative creative process. In *Iterate*, they

explain how to fail better. After laying out the four components of creative practice—intention, outcome, process, and evaluation—Sharp and Macklin describe iterative methods from a wide variety of fields. They show, for example, how Radiolab cohosts Jad Abumrad and Robert Krulwich experiment with radio as a storytelling medium; how professional skateboarder Amelia Bródka develops skateboarding tricks through trial and error; and how artistic polymath Miranda July explores human frailty through a variety of media and techniques. Whimsical illustrations tell parallel stories of iteration, as hard-working cartoon figures bake cupcakes, experiment with levitating office chairs, and think outside the box in toothbrush design (“let’s add propellers!”). All, in their various ways, use iteration to transform failure into creative outcomes. With *Iterate*, Sharp and Macklin offer useful lessons for anyone interested in the creative process. Case Studies: Allison Tauziet, winemaker; Matthew Maloney, animator; Jad Abumrad and Robert Krulwich, Radiolab cohosts; Wylie Dufresne, chef; Nathalie Pozzi, architect, and Eric Zimmerman, game designer; Andy Milne, jazz musician; Amelia Bródka, skateboarder; Baratunde Thurston, comedian; Cas Holman, toy designer; Miranda July, writer and filmmaker

Rust is a new systems programming language that combines the performance and low-level control of C and C++ with memory safety and thread safety. Rust’s modern, flexible types ensure your program is free of null pointer dereferences, double frees, dangling pointers, and similar bugs, all at compile time, without runtime overhead. In multi-threaded code, Rust catches data races at compile time, making concurrency much easier to use. Written by two experienced systems programmers, this book explains how Rust manages to bridge the gap between performance and safety, and how you can take advantage of it. Topics include: How Rust represents values in memory (with diagrams) Complete explanations of ownership, moves, borrows, and lifetimes Cargo, rustdoc, unit tests, and how to publish your code on crates.io, Rust’s public package repository High-level features like generic code, closures, collections, and iterators that make Rust productive and flexible Concurrency in Rust: threads, mutexes, channels, and atomics, all much safer to use than in C or C++ Unsafe code, and how to preserve the integrity of ordinary code that uses it Extended examples illustrating how pieces of the language fit together

[Copyright: 492ea19a53ed64af73cfaf28e36c3a79](#)