Bcpl The Language And Its Compiler

C is a favored and widely used programming language, particularly within the fields of science and engineering. C Programming for Scientists and Engineers with Applications guides readers through the fundamental, as well as the advanced concepts, of the C programming language as it applies to solving engineering and scientific problems. Ideal for readers with no prior programming experience, this text provides numerous sample problems and their solutions in the areas of mechanical engineering, electrical engineering, heat transfer, fluid mechanics, physics, chemistry, and more. It begins with a chapter focused on the basic terminology relating to hardware, software, problem definition and solution. From there readers are quickly brought into the key elements of C and will be writing their own code upon completion of Chapter 2. Concepts are then gradually built upon using a strong, structured approach with syntax and semantics presented in an easy-to-understand sentence format. Readers will find C Programming for Scientists and Engineers with Applications to be an engaging, user-friendly introduction to this popular language.
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
0805311912B04062001
Learning a language--any language--involves a process wherein you learn to rely less and less on instruction and more increasingly on the aspects of the language you've mastered. Whether you're learning French, Java, or C, at some point you'll set aside the tutorial and attempt to converse on your own. It's not necessary to know every subtle facet of French in order to speak it well, especially if there's a good dictionary available. Likewise, C programmers don't need to memorize every detail of C in order to write good programs. What they need instead is a reliable, comprehensive reference that they can keep nearby. C in a Nutshell is that reference. This long-awaited book is a complete reference to the C programming language and C runtime library. Its purpose is to serve as a convenient, reliable companion in your day-to-day work as a C programmer. C in a Nutshell covers virtually everything you need to program in C, describing all the elements of the language and illustrating their use with numerous examples. The book is divided into three distinct parts. The first part is a fast-paced description, reminiscent of the classic Kernighan & Ritchie text on which many C programmers cut their teeth. It focuses specifically on the C language and preprocessor directives, including extensions introduced to the ANSI standard in 1999. These topics and others are covered: Numeric constants Implicit and explicit type conversions Expressions and operators Functions Fixed-length and variable-length arrays Pointers Dynamic memory management Input and output The second part of the book is a comprehensive reference to the C runtime library; it includes an overview of the contents of the standard headers and a description of each standard library function. Part III provides the necessary knowledge of the C programmer's basic tools: the compiler, the make utility, and the debugger. The tools described here are those in the GNU software collection. C in a Nutshell is the perfect companion to K&R, and destined to be the most reached-for reference on your desk.

#1 NEW YORK TIMES BESTSELLER • Now a major motion picture directed by Steven Spielberg. “Enchanting . . . Willy Wonka meets The Matrix.”—USA Today • “As one adventure leads expertly to the next, time simply evaporates.”—Entertainment Weekly A world at stake. A quest for the ultimate prize. Are you ready? In the year 2045, reality is an ugly place. The only time Wade Watts really feels alive is when he’s jacked into the OASIS, a vast virtual world where most of humanity spends their days. When the eccentric creator of the OASIS dies, he leaves behind a series of fiendish puzzles, based on his obsession with the pop culture of decades past. Whoever is first to solve them will inherit his vast fortune—and control of the OASIS itself. Then Wade cracks the first clue. Suddenly he’s beset by rivals who’ll kill to take this prize. The race is on—and the only way to survive is to win. NAMED ONE OF THE BEST BOOKS OF THE YEAR BY Entertainment Weekly • San Francisco Chronicle • Village Voice • Chicago Sun-Times • iO9 • The AV Club “Delightful . . . the grown-up’s Harry Potter.”—HuffPost “An addictive read . . . part intergalactic scavenger hunt, part romance, and all heart.”—CNN “A most excellent ride . . . Cline stuffs his novel with a cornucopia of pop culture, as if to wink to the reader.”—Boston Globe “Ridiculously fun and large-hearted . . . Cline is that rare writer who can translate his own dorky enthusiasms into prose that’s both hilarious and compassionate.”—NPR “[A] fantastic page-turner . . . starts out like a simple bit of fun and winds up feeling like a rich and plausible picture of future friendships in a world not too distant from our own.”—iO9

Covers the nature of language, syntax, modeling objects, names, expressions, functions, control structures, global control, logic programming, representation and semantics of types, modules, generics, and domains

A comprehensive undergraduate textbook covering both theory and practical design issues, with an emphasis on object-oriented languages.

History of Programming Languages presents information pertinent to the technical aspects of the language design and creation. This book provides an understanding of the processes of language design as related to the environment in which languages are developed and the knowledge base available to the originators. Organized into 14 sections encompassing 77 chapters, this book begins with an overview of the programming techniques to use to help the system produce efficient programs. This text then discusses how to use parentheses to help the system identify identical subexpressions within an expression and thereby eliminate their duplicate calculation. Other chapters consider FORTRAN programming techniques needed to produce optimum object programs. This book discusses as well the developments leading to ALGOL 60. The final chapter presents the biography of Adin D. Falkoff. This book is a valuable resource for graduate students, practitioners, historians, statisticians, mathematicians, programmers, as well as computer scientists and specialists.

The title of this book contains the words ALGORITHMIC LANGUAGE, in the singular. This is meant to convey the idea that it deals not so much with the diversity of program ming languages, but rather with their commonalities. The task of formal program develop It allows classifying ment proved to be the ideal frame for demonstrating this unity. concepts and distinguishing fundamental notions from notational features; and it leads immediately to a systematic disposition. This approach is supported by didactic, practical, and theoretical considerations. The clarity of the structure of a programming language de signed according to the

principles of program transformation is remarkable. Of course there are various notations for such a language. The notation used in this book is mainly oriented towards ALGOL 68, but is also strongly influenced by PASCAL - it could equally well have been the other way round. In the appendices there are occa sional references to the styles used in ALGOL, PASCAL, LISP, and elsewhere.
In September 1918, World War I was nearing its end when Marguerite E. Harrison, a thirty-nine-year-old Baltimore socialite, wrote to the head of the U.S. Army's Military Intelligence Division asking for a job. The director asked for clarification. Did she mean a clerical position? No, she told him. She wanted to be a spy. Harrison, a member of a prominent Baltimore family, usually got her way. She had founded a school for sick children and wangled her way onto the staff of the Baltimore Sun. Fluent in four languages and knowledgeable of Europe, she was confident she could gather information for the U.S. government. The MID director agreed to hire her, and Marguerite Harrison became America's first female foreign intelligence officer. For the next seven years, she traveled to the world's most dangerous places—Berlin, Moscow, Siberia, and the Middle East—posing as a writer and filmmaker in order to spy for the U.S. Army and U.S. Department of State. With linguistic skills and knack for subterfuge, Harrison infiltrated Communist networks, foiled a German coup, located American prisoners in Russia, and probably helped American oil companies seeking entry into the Middle East. Along the way, she saved the life of King Kong creator Merian C. Cooper, twice survived imprisonment in Russia, and launched a women's explorer society whose members included Amelia Earhart and Margaret Mead. As incredible as her life was, Harrison has never been the subject of a published book-length biography. Past articles and chapters about her life relied heavily on her autobiography published in 1935, which omitted and distorted key aspects of her espionage career. This book draws on newly discovered documents in the U.S. National Archives, as well as Harrison's prison files in the archives of the Russian Federal Security Bureau in Moscow, Russia. Although Harrison portrayed herself as a writer who temporarily worked as a spy, this book documents that Harrison's espionage career was much more extensive and important than she revealed. She was one of America's most trusted agents in Germany, Russia and the Middle East after World War I when the United States sought to become a world power.
The authors provide clear examples and thorough explanations of every feature in the C language. They teach C vis-a-vis the UNIX operating system. A reference and tutorial to the C programming language. Annotation copyrighted by Book News, Inc., Portland, OR
The founder of Latina Rebels and a "Latinx Activist You Should Know"(Teen Vogue) arms women of color with the tools and knowledge they need to find success on their own terms For generations, Brown girls have had to push against powerful forces of sexism, racism, and classism, often feeling alone in the struggle. By founding Latina Rebels, Prisca Dorcas Mojica Rodríguez has created a community to help women fight together. In For Brown Girls with Sharp Edges and Tender Hearts, she offers wisdom and a liberating path forward for all women of color. She crafts powerful ways to address the challenges Brown girls face, from imposter syndrome to colorism. She empowers women to decolonize their worldview, and defy "universal" white narratives, by telling their own stories. Her book guides women of color toward a sense of pride and sisterhood and offers essential tools to

energize a movement. May it spark a fire within you.

""My name is Fire / but everyone calls me / Little Fire." In this beautiful, poetic ode to the invigorating power of fire, award-winning children's book author Jorge Argueta describes"¬,¬"in English, Spanish and Nahuat"¬,¬"the characteristics of fire from the perspective of one little flame. From its birth as a spark, Little Fire flits like a firefly and plays hide and seek inside a volcano. He grows between two sticks rubbed together or on a stone that strikes another. Little Fire is red, yellow, orange and turquoise. "I look like the sun / but I am no sun. / I am Fire, Little Fire / who laughs, / who dances." Little Fire sings, "sizzle, / hiss, / whoosh, / crackle, crackle." With stunningly beautiful illustrations by Felipe Ugalde Alcntara that depict the natural world, this poem about the importance of fire reflects Argueta's indigenous roots and his appreciation for nature. Containing the English and Spanish text on each page, the entire poem appears at the end in Nahuat, the language of Argueta's Pipil-Nahua ancestors. The sequel to Agua, Agita / Water, Little Water, this book is an excellent choice to encourage children to write their own poems about the environment. "

#1 NEW YORK TIMES BESTSELLER SELECTION OF THE REESE WITHERSPOON BOOK CLUB A HIGHLY ANTICIPATED, BEST BOOK OF SUMMER SELECTED BY * VOGUE * USA TODAY * ENTERTAINMENT WEEKLY * CNN * TOWN & COUNTRY * PARADE * BUSTLE * AND MORE! A “gripping” (Entertainment Weekly) mystery about a woman who thinks she’s found the love of her life—until he disappears. Before Owen Michaels disappears, he smuggles a note to his beloved wife of one year: Protect her. Despite her confusion and fear, Hannah Hall knows exactly to whom the note refers—Owen’s sixteen-year-old daughter, Bailey. Bailey, who lost her mother tragically as a child. Bailey, who wants absolutely nothing to do with her new stepmother. As Hannah’s increasingly desperate calls to Owen go unanswered, as the FBI arrests Owen’s boss, as a US marshal and federal agents arrive at her Sausalito home unannounced, Hannah quickly realizes her husband isn’t who he said he was. And that Bailey just may hold the key to figuring out Owen’s true identity—and why he really disappeared. Hannah and Bailey set out to discover the truth. But as they start putting together the pieces of Owen’s past, they soon realize they’re also building a new future—one neither of them could have anticipated. With its breakneck pacing, dizzying plot twists, and evocative family drama, The Last Thing He Told Me is a riveting mystery, certain to shock you with its final, heartbreaking turn.

BCPLThe Language and its CompilerCambridge University Press

I love virtual machines (VMs) and I have done for a long time.If that makes me "sad" or an "anorak", so be it. I love them because they are so much fun, as well as being so useful. They have an element of original sin (writing assembly programs and being in control of an entire machine), while still being able to claim that one is being a respectable member of the community (being structured, modular, high-level, object-oriented, and so on). They also allow one to design machines of one's own, unencumbered by the restrictions of a starts optimising it for some physical particular processor (at least, until one processor or other). I have been building virtual machines, on and off, since 1980 or there

abouts. It has always been something of a hobby for me; it has also turned out to be a technique of great power and applicability. I hope to continue working on them, perhaps on some of the ideas outlined in the last chapter (I certainly want to do some more work with register-based VMs and concur rency). I originally wanted to write the book from a purely semantic viewpoint.

BCPL is a simple systems programming language with a portable compiler that has been implemented on many machines from large mainframes to mini computers and microprocessors. The book provides an introduction to the language, paying particular attention to programming style. In addition, it covers the more machine-independent parts of the BCPL library and outlines various debugging aids that most implementations provide. The syntax analysis phase of the compiler is described in detail, giving a realistic example of a typical application of the language. This and other substantial examples given in the book will be of interest both to serious users of BCPL and to computer writers. There is a chapter concerned with the portability code generator design. The reference for BCPL appears as the final chapter.

Little Lobo and Bernabé are back in this joyful story about coming together and celebrating community, a lively follow-up to ¡Vamos! Let's Go Eat, by Pura Belpré Medal–winning illustrator Raúl the Third. People are always crossing the bridge for work, to visit family, or for play. Some going this way; others going that way. Back and forth they go. With friends on foot and in bicycles, in cars and trucks, the bridge is an incredibly busy place with many different types of vehicles. Little Lobo and his dog Bernabé have a new truck and they are using it to carry party supplies over the bridge with their pals El Toro and La Oink Oink. The line is long and everyone on the bridge is stuck. How will they pass the time? Eventually everyone comes together for an epic party on the bridge between two different countries. Richard Scarry's Cars and Trucks and Things That Go gets Mexican American makeover in this joyful story about coming together.

Shedding profound natural light on the inner lives of migrant workers, Jaime Cortez's debut collection ushers in a new era of American literature that gives voice to a marginalized generation of migrant workers in the West. The first-ever collection of short stories by Jaime Cortez, Gordo is set in a migrant workers camp near Watsonville, California in the 1970s. A young, probably gay, boy named Gordo puts on a wrestler's mask and throws fists with a boy in the neighborhood, fighting his own tears as he tries to grow into the idea of manhood so imposed on him by his father. As he comes of age, Gordo learns about sex, watches his father's drunken fights, and discovers even his own documented Mexican-American parents are wary of illegal migrants. Fat Cookie, high schooler and resident artist, uses tiny library pencils to draw huge murals of graffiti flowers along the camp's blank walls, the words "CHICANO POWER" boldly lettered across, until she runs away from home one day with her mother's boyfriend, Manny, and steals her mother's Panasonic radio for a final dance competition among the camp kids before she disappears. And then there are Los

Tigres, the perfect pair of twins so dark they look like indios, Pepito and Manuel, who show up at Gyrich Farms every season without fail. Los Tigres, champion drinkers, end up assaulting each other in a drunken brawl, until one of them is rushed to the emergency room still slumped in an upholstered chair tied to the back of a pick-up truck. These scenes from Steinbeck Country seen so intimately from within are full of humor, family drama, and a sweet frankness about serious matters – who belongs to America and how are they treated? How does one learn decency, when laborers, grown adults, must fear for their lives and livelihoods as they try to do everything to bring home a paycheck? Written with balance and poise, Cortez braids together elegant and inviting stories about life on a California camp, in essence redefining what all-American means.

Temporal logic is gaining recognition as an attractive and versatile formalism for rigorously specifying and reasoning about computer programs, digital circuits and message-passing systems. This book introduces Tempura, a programming language based on temporal logic, Tempura provides a way of directly executing suitable temporal logic specifications of digital circuits, parallel programs and other dynamic systems. Since every Tempura statement is also a temporal formula, the entire temporal logic formalism can be used as the assertion language and semantics. One result is that Tempura has the two seemingly contradictory properties of being a logic programming language and having imperative constructs such as assignment statements. The presentation investigates Interval Temporal Logic, a formalism with conventional temporal operators such as next and always as well as lesser known ones such as chop. This provides the basis for Tempura. The design of an interpreter for Tempura is also included, as are a variety of sample Tempura programs illustrating how to model both hardware and software.

Ever since he was a boy, John Brown had hated slavery. He was an abolitionist, a person who believed that no one should be able to own others. Many abolitionists hope that strong words would convince people to end slavery, but John thought words were not enough. He was determined to fight-even if it meant death. In John Brown, author Tom Streissguth and illustrator Ralph L. Ramstad capture the fiery determination of the man whose actions helped to bring about the Civil War.

C (/si?/, as in the letter c) is a general-purpose, procedural computer programming language supporting structured programming, lexical variable scope, and recursion, while a static type system prevents unintended operations. By design, C provides constructs that map efficiently to typical machine instructions and has found lasting use in applications previously coded in assembly language. Such applications include operating systems and various application software for computers, from supercomputers to embedded systems.C was originally developed at Bell Labs by Dennis Ritchie between 1972 and 1973 to make utilities running on Unix. Later, it was applied to re-implementing the kernel of the Unix

operating system.[6] During the 1980s, C gradually gained popularity. Nowadays, it is one of the most widely used programming languages, [7][8] with C compilers from various vendors available for the majority of existing computer architectures and operating systems. C has been standardized by the ANSI since 1989 (see ANSI C) and by the International Organization for Standardization.C is an imperative procedural language. It was designed to be compiled using a relatively straightforward compiler to provide low-level access to memory and language constructs that map efficiently to machine instructions, all with minimal runtime support. Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant C program written with portability in mind can be compiled for a wide variety of computer platforms and operating systems with few changes to its source code. The language is available on various platforms, from embedded microcontrollers to supercomputers.The origin of C is closely tied to the development of the Unix operating system, originally implemented in assembly language on a PDP-7 by Dennis Ritchie and Ken Thompson, incorporating several ideas from colleagues. Eventually, they decided to port the operating system to a PDP-11. The original PDP-11 version of Unix was also developed in assembly language.[10]Thompson desired a programming language to make utilities for the new platform. At first, he tried to make a Fortran compiler but soon gave up the idea. Instead, he created a cut-down version of the recently developed BCPL systems programming language. The official description of BCPL was not available at the time, [11] and Thompson modified the syntax to be less wordy, producing the similar but somewhat simpler B.[10] However, few utilities were ultimately written in B because it was too slow, and B could not take advantage of PDP-11 features such as byte addressability.In 1972, Ritchie started to improve B, which resulted in creating a new language C.[12] The C compiler and some utilities made with it were included in Version 2 Unix.[13]At Version 4 Unix released at Nov. 1973, the Unix kernel was extensively re-implemented by C.[10] By this time, the C language had acquired some powerful features such as struct types.Unix was one of the first operating system kernels implemented in a language other than assembly. Earlier instances include the Multics system (which was written in PL/I) and Master Control Program (MCP) for the Burroughs B5000 (which was written in ALGOL) in 1961. In around 1977, Ritchie and Stephen C. Johnson made further changes to the language to facilitate portability of the Unix operating system. Johnson's Portable C Compiler served as the basis for several implementations of C on new platforms.[12]Many later languages have borrowed directly or indirectly from C, including C++, C#, Unix's C shell, D, Go, Java, JavaScript, Limbo, LPC, Objective-C, Perl, PHP, Python, Rust, Swift, Verilog and SystemVerilog (hardware description languages).[5] These languages have drawn many of their control structures and other basic features from C. Most of them (Python being a dramatic exception) also express highly similar syntax to

It's been four years since Monica disrupted the Cinque household, but starting trouble is like riding a bike for Monica, and she's pedaling full steam ahead. The foundation of Jasmine and James Cinque's marriage has been shaky ever since they dared to bring a third person into their union. Now they're trying to repair the damage they've done, to regain trust and repair broken hearts. But with so much drama in their past, it won't be hard for Monica to come in and shake things up a little. If Monica is smart, though, she'll watch her back. With so many enemies gunning to take her down, Monica has to decide if Philly is where she wants to be or if she should run back to the ATL where it's safe. Or is it? Breaking up is hard to do, but making up proves to be just as challenging, especially if Monica has anything to do with it.

To construct a compiler for a modern higher-level programming languagel one needs to structure the translation to a machine-like intermediate language in a way that reflects the semantics of the language. little is said about such struc turing in compiler texts that are intended to cover a wide variety of program ming languages. More is said in the literature on semantics-directed compiler construction [1] but here too the viewpoint is very general (though limited to 1 languages with a finite number of syntactic types). On the other handl there is a considerable body of work using the continuation-passing transformation to structure compilers for the specific case of call-by-value languages such as SCHEME and ML [21 3]. ln this paperl we will describe a method of structuring the translation of ALGOL-like languages that is based on the functor-category semantics devel oped by Reynolds [4] and Oles [51 6]. An alternative approach using category theory to structure compilers is the early work of F. L. Morris [7]1 which anticipates our treatment of boolean expressionsl but does not deal with procedures. 2 Types and Syntax An ALGOL-like language is a typed lambda calculus with an unusual repertoire of primitive types. Throughout most of this paper we assume that the primi tive types are comm(and) int(eger)exp(ression) int(eger)acc(eptor) int(eger)var(iable) I and that the set 8 of types is the least set containing these primitive types and closed under the binary operation -.

Ahoy mateys! A young squirrel has always dreamed of sailing the seas as a pirate. So when he finds a treasure map, he can't believe his luck! An X marks the spot of 100,000 NUTS! Set sail for adventure with vibrant illustrations and imaginative rhyming text to discover how exciting a pirate's life can be! The Capstone Interactive edition comes with simultaneous access for every student in your school and includes read aloud audio recorded by professional voice over artists.

This book uses a functional programming language (F#) as a metalanguage to present all concepts and examples, and thus has an operational flavour, enabling practical experiments and exercises. It includes basic concepts such as abstract syntax, interpretation, stack machines, compilation, type checking, garbage collection, and real machine code. Also included are more advanced topics on polymorphic types, type inference using unification, co- and contravariant

types, continuations, and backwards code generation with on-the-fly peephole optimization. This second edition includes two new chapters. One describes compilation and type checking of a full functional language, tying together the previous chapters. The other describes how to compile a C subset to real (x86) hardware, as a smooth extension of the previously presented compilers.The examples present several interpreters and compilers for toy languages, including compilers for a small but usable subset of C, abstract machines, a garbage collector, and ML-style polymorphic type inference. Each chapter has exercises. Programming Language Concepts covers practical construction of lexers and parsers, but not regular expressions, automata and grammars, which are well covered already. It discusses the design and technology of Java and C# to strengthen students' understanding of these widely used languages.

Three former CIA officers share their techniques for lie detection, outlining methods for identifying deceptiveness as revealed by verbal and non-verbal behaviors from facial expressions and grooming gestures to invoking religion and using qualifying language.

Do you wake up dreading the day? Do you feel ciscouraged with what you've accomplished in life? Do you want greater self-esteem, productivity, and joy in daily living? If so, you will benefit from this revolutionary way of brightening your moods without drugs or lengthy therapy. All you need is your own common sense and the easy-to-follow methods revealed in this book by one of the country's foremost authorities on mood and personal relationship problems. In Ten Days to Self-esteem, Dr. David Burns presents innovative, clear, and compassionate methods that will help you identify the causes of your mood slumps and develop a more positive outlook on life. You will learn that You feel the way you think: Negative feelings like guilt, anger, and depression do not result from the bad things that happen to you, but from the way you think about these events. This simple but revolutionary idea can change your life! You can change the way you feel: You will discover why you get depressed and learn how to brighten your outlook when you're in a slump. You can enjoy greater happiness, productivity, and intimacy—without drugs or lengthy therapy. Can a self-help book do all this? Studies show that two thirds of depressed readers of Dr. Burns's classic bestseller, Feeling Good: The New Mood Therapy, experienced dramatic felief in just four weeks without psychotherapy or antidepressant medications. Three-year follow-up studies revealed that readers did not relapse but continued to enjoy their positive outlook. Ten Days to Self-esteem offers a powerful new tool that provides hope and healing in ten easy steps. The methods are based on common sense and are not difficult to apply. Research shows that they really work! Feeling good feels wonderful. You owe it to yourself to feel good!

This book describes the evolution of computer science in the form of seven overlapping, intermingling, parallel histories that unfold concurrently in the course of the two decades. Author Subrata Dasgupta named the two decades from 1970 to 1990 as the second age of

computer science to distinguish it from the preceding genesis of the science and the age of the Internet/World Wide Web that followed--
The Dictionary of e-business: * Now includes extended coverage of wireless and mobile terms * Is authored by an expert in the field * Presents more than 350 new entries on Java, XML, Customer Relationship Management, mCommerce and more technical language of eBusiness (e.g. security) * Demontrates clear applications to both technical and business markets * Covers all the latest developments in this fast moving field
The earth, viewed through the window of an airplane, shows a regularity and reptition of features, for example, hills, valleys, rivers, lakes, and forests. Nevertheless, there is great local variation; Vermont does not look like Utah. Similarly, if we rise above the details of a few programming languages, we can discern features that are common to many languages. This is the programming language landscape; the main features include variables, types, control structures, and input/output. Again, there is local variation; Pascal does not look like Basic. This work is a broad and comprehensive discussion of the principal features of the major programming languages. A Study of Concepts The text surveys the landscape of programming languages and its features. Each chapter concentrates on a single language concept. A simple model of the feature, expressed as a mini-language, is presented. This allows us to study an issue in depth and relative isolation. Each chapter concludes with a discussion of the way in which the concept is incorporated into some well-known languages. This permits a reasonably complete coverage of language issues.
Best Book of 2020 New York Times |NPR | New York Post "This hushed suspense tale about thwarted dreams of escape may be her best one yet . . . Its own kind of masterpiece." --Maureen Corrigan, The Washington Post "A new Tana French is always cause for celebration . . . Read it once for the plot; read it again for the beauty and subtlety of French's writing." --Sarah Lyall, The New York Times Cal Hooper thought a fixer-upper in a bucolic Irish village would be the perfect escape. After twenty-five years in the Chicago police force and a bruising divorce, he just wants to build a new life in a pretty spot with a good pub where nothing much happens. But when a local kid whose brother has gone missing arm-twists him into investigating, Cal uncovers layers of darkness beneath his picturesque retreat, and starts to realize that even small towns shelter dangerous secrets. "One of the greatest crime novelists writing today" (Vox) weaves a masterful, atmospheric tale of suspense, asking how to tell right from wrong in a world where neither is simple, and what we stake on that decision.
The Art of UNIX Programming poses the belief that understanding the unwritten UNIX engineering tradition and mastering its design patterns will help programmers of all stripes to become better programmers. This book attempts to capture the engineering wisdom and design philosophy of the UNIX, Linux, and Open Source software development community as it has evolved over the past three decades, and as it is applied today by the most experienced programmers. Eric Raymond offers the next generation of "hackers" the unique opportunity to learn the connection between UNIX philosophy and practice through careful case studies of the very best UNIX/Linux programs.