Lex Yacc Levine 2nd Edition

Introduction to Compilers and Language DesignMySQL and MSQLPorting UNIX SoftwareC++ In a NutshellModern Compiler DesignEngineering a CompilerOracle Performance Tuning□□□□□□High Performance ComputingCompiler Design Using FLEX and YACClex & yaccThe Go Programming LanguageHead First GoPractical C++ ProgrammingFlex & BisonForthcoming BooksMicrosoft RPC Programming GuideGuide to Writing DCE ApplicationsModern Compiler Implementation in CUNIX Systems Programming for SVR4TCP/IP Network AdministrationThe Definitive ANTLR 4 ReferenceLearning GNU EmacsDigital Signal Processing with Field Programmable Gate ArraysProgramming with GNU SoftwareEffective awk ProgrammingFlex & BisonMetaprogramming GPUs with ShLinux in a NutshellPractical Color ManagementBuilding a successful software businessSoftware Portability with ImakeThe Pragmatic ProgrammerModern Compiler Implementation in JavaSed and Awk Pocket ReferenceText Processing in PythonProgramming with CursesLPI Linux Certification in a NutshellBisonUNIX for FORTRAN Programmers

Introduction to Compilers and Language Design

This entirely revised second edition of Engineering a Compiler is full of technical updates and new material covering the latest developments in compiler technology. In this comprehensive text you will learn important techniques for constructing a modern compiler. Leading educators and researchers Keith Cooper and Linda Torczon combine basic principles with pragmatic insights from their experience building state-of-the-art compilers. They will help you fully understand important techniques such as compilation of imperative and object-oriented languages, construction of static single assignment forms, instruction scheduling, and graph-coloring register allocation. In-depth treatment of algorithms and techniques used in the front end of a modern compiler Focus on code optimization and code generation, the primary areas of recent research and development Improvements in presentation including conceptual overviews for each chapter, summaries and review questions for sections, and prominent placement of definitions for new terms Examples drawn from several different programming languages

MySQL and MSQL

If you need to parse or process text data in Linux or Unix, this useful book explains how to use flex and bison to solve your problems quickly. flex & bison is the long-awaited sequel to the classic O'Reilly book, lex & yacc. In the nearly two decades since the original book was published, the flex and bison utilities have proven to be more reliable and more powerful than the original Unix tools. flex & bison covers the same core functionality vital to Linux and Unix program development, along with several important new topics. You'll find revised tutorials for novices and references for advanced users, as well as an explanation of each utility's basic usage and simple, standalone applications you can create with them. With flex & bison, you'll discover the wide range of uses these flexible tools offer. Address syntax crunching that regular expressions tools can't handle Build compilers and interpreters, and handle a wide range of text processing functions Interpret code, configuration files, or any other structured format Learn key programming techniques, including abstract syntax trees and symbol tables Implement a full SQL grammar-with complete sample code Use new features such as pure (reentrant) lexers and parsers, powerful GLR parsers, and interfaces to C++

Porting UNIX Software

TCP/IP Network Administration is a complete guide to setting up and running a TCP/IP network for administrators of networks of systems or users of home systems that access the Internet. It starts with the fundamentals: what the protocols do and how they work, how to request a network address and a name (the forms needed are included in an appendix), and how to set up your network.Beyond basic setup, the book discusses how to configure important network applications, including sendmail, the r* commands, and some simple setups for NIS and NFS. There are also chapters on troubleshooting and security. In addition, this book covers several important packages that are available from the Net (such as gated).Contents include: Overview of TCP/IP Delivering the data Name service concepts Getting started Basic configuration Configuring the interface Configuring routing Configuring DNS name service Network applications sendmail Troubleshooting TCP/IP Network security Other sources of information Appendixes include: network contacts, forms, a gated reference, named reference Covers BSD and System V TCP/IP implementations.

C++ In a Nutshell

Covering the LPI General Linux Exams 101 and 102, this helpful test preparation guidebook offers a detailed summary of each exam, along with hands-on exercises, extensive explanations and review, and practice exams. Original. (Intermediate/Advanced)

Modern Compiler Design

This updated classic teaches UNIX and Linux developers how to solve specific problems by processing text data, using two key UNIX utilities--Flex and Bison. Covers key programming techniques, including syntax trees and symbol tables. 300 pp

Engineering a Compiler

bull; Demonstrates how Python is the perfect language for text-processing functions. bull; Provides practical pointers and tips that emphasize efficient, flexible, and maintainable approaches to text-processing challenges. bull; Helps programmers develop solutions for dealing with the increasing amounts of data with which we are all inundated.

Oracle Performance Tuning

This new, expanded textbook describes all phases of a modern compiler: lexical analysis, parsing, abstract syntax, semantic actions, intermediate representations, instruction selection via tree matching, dataflow analysis, graph-coloring register allocation, and runtime systems. It includes good coverage of current techniques in code generation and register allocation, as well as functional and object-oriented languages, that are missing from most books. In addition, more advanced chapters are now included so that it can be used as the basis for a two-semester or graduate course. The most accepted and successful techniques are described in a concise way, rather than as an exhaustive catalog of every possible variant. Detailed descriptions of the interfaces between modules of a compiler are illustrated with actual C header files. The first part of the book, Fundamentals of Compilation, is suitable for a one-semester first course in compiler design. The second part, Advanced Topics, which includes the advanced chapters, covers the compilation of object-oriented and functional languages, garbage collection, loop optimizations, SSA form, loop scheduling, and optimization for cache-memory hierarchies.

□□□□□□□

This book is a high-level overview of Sh and its relationship to other realtime shading and Graphics processing unit programming languages. It is a reference manual and language specification and methodically and exhaustively presents details of the various features of Sh.

High Performance Computing

An introduction to GNU Emacs editos, one of the most widely used and powerful editors available under UNIX. Provides a solid introduction to basic editing, a look at several important "editing modes", and a brief introduction to customization and Emacs LISP programming.

Compiler Design Using FLEX and YACC

C++ is a powerful, highly flexible, and adaptable programming language that allows software engineers to organize and process information quickly and effectively. But this high-level language is relatively difficult to master, even if you already know the C programming language.The new second edition of "Practical C++ Programming is a complete introduction to the C++ language for programmers who are learning C++. Reflecting the latest changes to the C++ standard, this new edition takes a useful down-to-earth approach, placing a strong emphasis on how to design clean, elegant code.In short, to-the-point chapters, all aspects of programming are covered including style, software engineering, programming design, object-oriented design, and debugging.It also covers common mistakes and how to find (and avoid) them. End of chapter exercises help you ensure you've mastered the material. Steve Oualline's clear, easy-going writing style and hands-on approach to learning make "Practical C++ Programming a nearly painless way to master this complex but powerful programming language.

lex & yacc

Programmers run into parsing problems all the time. Whether it's a data format like

JSON, a network protocol like SMTP, a server configuration file for Apache, a PostScript/PDF file, or a simple spreadsheet macro language--ANTLR v4 and this book will demystify the process. ANTLR v4 has been rewritten from scratch to make it easier than ever to build parsers and the language applications built on top. This completely rewritten new edition of the bestselling Definitive ANTLR Reference shows you how to take advantage of these new features. Build your own languages with ANTLR v4, using ANTLR's new advanced parsing technology. In this book, you'll learn how ANTLR automatically builds a data structure representing the input (parse tree) and generates code that can walk the tree (visitor). You can use that combination to implement data readers, language interpreters, and translators. You'll start by learning how to identify grammar patterns in language reference manuals and then slowly start building increasingly complex grammars. Next, you'll build applications based upon those grammars by walking the automatically generated parse trees. Then you'll tackle some nasty language problems by parsing files containing more than one language (such as XML, Java, and Javadoc). You'll also see how to take absolute control over parsing by embedding Java actions into the grammar. You'll learn directly from well-known parsing expert Terence Parr, the ANTLR creator and project lead. You'll master ANTLR grammar construction and learn how to build language tools using the built-in parse tree visitor mechanism. The book teaches using real-world examples and shows you how to use ANTLR to build such things as a data file reader, a JSON to XML translator, an R parser, and a Java class->interface extractor. This book is your ticket to becoming a parsing guru! What You Need: ANTLR 4.0 and above. Java development tools. Ant build system optional(needed for building ANTLR from source)

The Go Programming Language

There are many complex issues surrounding the use of the Japanese language in computing. This book provides detailed information on all aspects of handling Japanese text on computer systems. It tries to bring all of the relevant information together in a single book--covering everything from the origins of modern-day Japanese to the latest information on specific emerging computer encoding standards.

Head First Go

Effective awk Programming,3rd Edition, focuses entirely on awk, exploring it in the greatest depth of the three awk titles we carry. It's an excellent companion piece to the more broadly focused second edition. This book provides complete coverage of the gawk 3.1 language as well as the most up-to-date coverage of the POSIX standard for awk available anywhere. Author Arnold Robbins clearly distinguishes standard awk features from GNU awk (gawk)-specific features, shines light into many of the "dark corners" of the language (areas to watch out for when programming), and devotes two full chapters to example programs. A brand new chapter is devoted to TCP/IP networking with gawk. He includes a summary of how the awk language evolved. The book also covers: Internationalization of gawk Interfacing to i18n at the awk level Two-way pipes TCP/IP networking via the two-way pipe interface The new PROCINFO array, which provides information about running gawk Profiling and pretty-printing awk programs In addition to covering the awk language, this book serves as the official "User's Guide" for the GNU implementation of awk (gawk), describing in an integrated fashion the extensions available to the System V Release 4 version of awk that are also available in gawk. As the official gawk User's Guide, this book will also be available electronically, and can be freely copied and distributed under the terms of the Free Software Foundation's Free Documentation License (FDL). A portion of the proceeds from sales of this book will go to the Free Software Foundation to support further development of free and open source software. The third edition of Effective awk Programming is a GNU Manual and is published by O'Reilly & Associates under the Free Software Foundation's Free Documentation License (FDL). A portion of the proceeds from the sale of this book is donated to the Free Software Foundation to further development of GNU software. This book is also available in electronic form; you have the freedom to modify this GNU Manual, like GNU software. Copies published by the Free Software Foundation raise funds for GNU development.

Practical C++ Programming

Here is a complete package for programmers who are new to UNIX or who would like to make better use of the system. The book provides an introduction to all the tools needed for a C programmer. The CD contains sources and binaries for the most popular GNU tools, including their C/C++ compiler.

Flex & Bison

This handy book is an indispensable reference to information presented in O'Reilly's larger volumes, and is a concise summary of regular expressions and pattern matching.

Forthcoming Books

This new Nutshell handbook--the only book available on IMAKE--is ideal for X and UNIX programmers who want their software to be portable. The first section is a general explanation of IMAKE, X configuration files, and how to write and debug IMAKE files. The second section describes how to write configuration files and presents a configuration file architecture that allows development of coexisting sets of configuration files. Several sample sets of configuration files are described and are available free over the net.

Microsoft RPC Programming Guide

UNIX for FORTRAN Programmers provides the serious scientific programmer with an introduction to the UNIX operating system and its tools. The intent of the book is to minimize the UNIX entry barrier and to familiarize readers with the most important tools so they can be productive as quickly as possible. UNIX for FORTRAN Programmers shows readers how to do things they're interested in: not just how to use a tool such as make or rcs, but how to use it in program development and how it fits into the toolset as a whole. The tools discussed include: The FORTRAN compiler (f77). UNIX interactive command languages, or shells (csh for interactive use, sh for shell programming). vi, the standard UNIX editor. Object library management tools (ar and ranlib). The programming environment (I/O, basic system calls, error handling). The adb and dbx debuggers. prof, gprof, time, profiling tools. make, a tool for automating complex compilations. rcs, a source code management system for large projects. Common porting problems.

Guide to Writing DCE Applications

Contains an introduction to the operating system with detailed documentation on commands, utilities, programs, system configuration, and networking.

Modern Compiler Implementation in C

A guide to the SQL-based database applications covers installation, configuration, interfaces, and administration

UNIX Systems Programming for SVR4

C++ is a powerful, highly flexible, and adaptable programming language that allows software engineers to organize and process information quickly and effectively. This is a complete reference to C++.

TCP/IP Network Administration

If you work with computers, you owe it to yourself to understand the new directions that workstation architecture has taken in the last half decade. This book covers everything, from the basics of modern workstation architecture to structuring benchmarks to squeezing more performance out of critical applications. Explains how optimizing compilers work; discusses what a good compiler can and can't do; looks at the high-performance future; discusses several of the "standard" industry benchmarks; and more.

The Definitive ANTLR 4 Reference

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.

Learning GNU Emacs

"Modern Compiler Design" makes the topic of compiler design more accessible by focusing on principles and techniques of wide application. By carefully distinguishing between the essential (material that has a high chance of being useful) and the incidental (material that will be of benefit only in exceptional cases) much useful information was packed in this comprehensive volume. The student who has finished this book can expect to understand the workings of and add to a language processor for each of the modern paradigms, and be able to read the literature on how to proceed. The first provides a firm basis, the second potential for growth.

Digital Signal Processing with Field Programmable Gate Arrays

What others in the trenches say about The Pragmatic Programmer “The cool thing about this book is that it’s great for keeping the programming process fresh. The book helps you to continue to grow and clearly comes from people who have been there.” —Kent Beck, author of Extreme Programming Explained: Embrace Change “I found this book to be a great mix of solid advice and wonderful analogies!” —Martin Fowler, author of Refactoring and UML Distilled “I would buy a copy, read it twice, then tell all my colleagues to run out and grab a copy. This is a book I would never loan because I would worry about it being lost.” —Kevin Ruland, Management Science, MSG-Logistics “The wisdom and practical experience of the authors is obvious. The topics presented are relevant and useful. By far its greatest strength for me has been the outstanding analogies—tracer bullets, broken windows, and the fabulous helicopter-based explanation of the need for orthogonality, especially in a crisis situation. I have little doubt that this book will eventually become an excellent source of useful information for journeymen programmers and expert mentors alike.” —John Lakos, author of Large-Scale C++ Software Design “This is the sort of book I will buy a dozen copies of when it comes out so I can give it to my clients.” —Eric Vought, Software Engineer “Most modern books on software development fail to cover the basics of what makes a great software developer, instead spending their time on syntax or technology where in reality the greatest leverage possible for any software team is in having talented developers who really know their craft well. An excellent book." —Pete McBreen, Independent Consultant "Since reading this book, I have implemented many of the practical suggestions and tips it contains. Across the board, they have saved my company time and money while helping me get my job done quicker! This should be a desktop reference for everyone who works with code for a living." —Jared Richardson, Senior Software Developer, iRenaissance, Inc. "I would like to see this issued to every new employee at my company." —Chris Cleeland, Senior Software Engineer, Object Computing, Inc. "If I'm putting together a project, it's the authors of this book that I want. . . . And failing that I'd settle for people who've read their book." —Ward Cunningham Straight from the programming trenches, The Pragmatic Programmer cuts through the increasing specialization and technicalities of modern software development to examine the core process--taking a requirement and producing working, maintainable code that delights its users. It covers topics ranging from personal responsibility and career development to architectural techniques for keeping your code flexible and easy to adapt and reuse. Read this book, and you'll learn how to Fight software rot; Avoid the trap of duplicating knowledge; Write flexible, dynamic, and adaptable code; Avoid programming by coincidence; Bullet-proof your code with contracts, assertions, and exceptions; Capture real requirements; Test ruthlessly and effectively; Delight your users; Build teams of pragmatic programmers; and Make your developments more precise with automation. Written as a series of self-contained sections and filled with entertaining anecdotes, thoughtful examples, and interesting analogies, The Pragmatic Programmer illustrates the best practices and major pitfalls of many different aspects of software development. Whether you're a new coder, an experienced programmer, or a manager responsible for software projects, use these lessons daily, and you'll quickly see improvements in personal productivity, accuracy, and job satisfaction. You'll learn skills and develop habits and attitudes that form the foundation for long-term success in your career. You'll become a Pragmatic Programmer.

Programming with GNU Software

Starts with an overview of today's FPGA technology, devices, and tools for designing state-of-the-art DSP systems. A case study in the first chapter is the basis for more than 30 design examples throughout. The following chapters deal with computer arithmetic concepts, theory and the implementation of FIR and IIR filters, multirate digital signal processing systems, DFT and FFT algorithms, and advanced algorithms with high future potential. Each chapter contains exercises. The VERILOG source code and a glossary are given in the appendices, while the accompanying CD-ROM contains the examples in VHDL and Verilog code as well as the newest Altera "Baseline" software. This edition has a new chapter on adaptive filters, new sections on division and floating point arithmetics, an up-date to the current Altera software, and some new exercises.

Effective awk Programming

The ORACLE relational database management system is the most popular database system in use today. ORACLE offers tremendous power and flexibility, but at some cost. Demands for fast response make performance a major issue. Whether you're a manager, a designer, a programmer, or an administrator, with the tips presented here, you can dramatically increase the performance of your ORACLE system--and save time and bother. 9/93.

Flex & Bison

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

Metaprogramming GPUs with Sh

The first book to deal with the whole life cycle of porting, from obtaining software to building the documentation, Porting UNIX Software offers complete coverage of porting issues, including how to obtain and load the software and make changes in programs to get them working. Includes summaries of major UNIX features that vary between systems.

Linux in a Nutshell

This textbook describes all phases of a compiler: lexical analysis, parsing, abstract syntax, semantic actions, intermediate representations, instruction selection via tree matching, dataflow analysis, graph-coloring register allocation, and runtime systems. It includes good coverage of current techniques in code generation and register allocation, as well as the compilation of functional and object-oriented languages, that is missing from most books. The most accepted and successful techniques are described concisely, rather than as an exhaustive catalog of every possible variant, and illustrated with actual Java classes. This second edition has been extensively rewritten to include more discussion of Java and object-oriented programming concepts, such as visitor patterns. A unique feature is the newly redesigned compiler project in Java, for a subset of Java itself. The project includes both front-end and back-end phases, so that students can build a complete working compiler in one semester.

Practical Color Management

A compiler translates a program written in a high level language into a program written in a lower level language. For students of computer science, building a compiler from scratch is a rite of passage: a challenging and fun project that offers insight into many different aspects of computer science, some deeply theoretical, and others highly practical. This book offers a one semester introduction into compiler construction, enabling the reader to build a simple compiler that accepts a C-like language and translates it into working X86 or ARM assembly language. It is most suitable for undergraduate students who have some experience programming in C, and have taken courses in data structures and computer architecture.

Building a successful software business

This book is a comprehensive practical guide to the design, development, programming, and construction of compilers. It details the techniques and methods used to implement the different phases of the compiler with the help of FLEX and YACC tools. The topics in the book are systematically arranged to help students understand and write reliable programs in FLEX and YACC. The uses of these tools are amply demonstrated through more than a hundred solved programs to facilitate a thorough understanding of theoretical implementations discussed. KEY FEATURES l Discusses the theory and format of Lex specifications and describes in detail the features and options available in FLEX. l Emphasizes the different YACC programming strategies to check the validity of the input source program. l Includes detailed discussion on construction of different phases of compiler such as Lexical Analyzer, Syntax Analyzer, Type Checker, Intermediate Code Generation, Symbol Table, and Error Recovery. l Discusses the Symbol Table implementation—considered to be the most difficult phase to implement—in an utmost simple manner with examples and illustrations. l Emphasizes Type Checking phase with illustrations. The book is primarily designed as a textbook to serve the needs of B.Tech. students in computer science and engineering as well as those of MCA students for a course in Compiler Design Lab.

Software Portability with Imake

Bison is a general-purpose parser generator that converts an annotated context-free grammar into a deterministic LR or generalized LR (GLR) parser employing LALR(1) parser tables. As an experimental feature, Bison can also generate IELR(1) or canonical LR(1) parser tables. Once you are proficient with Bison, you can use it to develop a wide range of language parsers, from those used in simple desk calculators to complex programming languages. Bison is upward compatible with Yacc: all properly-written Yacc grammars ought to work with Bison with no change. Anyone familiar with Yacc should be able to use Bison with little trouble. You need to be fluent in C or C++ programming in order to use Bison or to understand this manual. Java is also supported as an experimental feature. We begin with tutorial chapters that explain the basic concepts of using Bison and show three explained examples, each building on the last. If you don't know Bison or Yacc, start by reading these chapters. Reference chapters follow, which describe specific aspects of Bison in detail. Bison was written originally by Robert Corbett. Richard Stallman made it Yacc-compatible. Wilfred Hansen of Carnegie Mellon University added multi-character string literals and other features. Since then, Bison has grown more robust and evolved many other new features thanks to the hard work of a long list of volunteers. This edition corresponds to version 3.0.4 of Bison.

The Pragmatic Programmer

Modern Compiler Implementation in Java

What will you learn from this book? Go makes it easy to build software that’s simple, reliable, and efficient. Andthis book makes it easy for programmers like you to get started. Googledesigned Go for high-performance networking and multiprocessing, but—like Python and JavaScript—the language is easy to read and use. With thispractical hands-on guide, you’ll learn how to write Go code using clearexamples that demonstrate the language in action. Best of all, you’ll understandthe conventions and techniques that employers want entry-level Godevelopers to know. Why does this book look so different? Based on the latest research in cognitive science and learning theory, HeadFirst Go uses a visually rich format to engage your mind rather than a textheavyapproach that puts you to sleep. Why waste your time struggling withnew concepts? This multisensory learning experience is designed for theway your brain really works.

Sed and Awk Pocket Reference

The expanding global market offers many opportunities for the software industry; however, many new software companies never realize their potential. They write some great code--but they can't address the "business" side of running a profitable enterprise. Many potentially great companies have fallen by the wayside because their founders didn't understand their market, didn't understand how to get the word out, or didn't understand the mechanics of the business. Building a Successful Software Business is a handbook for the new software entrepreneur and the old hand alike. If you're thinking of starting a company around a program you've written, this book will guide you toward success. If you're an old hand in the software industry, this book will help you sharpen your skills or will provide a refresher course. If you're thinking of building a company around some software you've developed, there's no better time than the present. Let this book start you on the way to success. Topics include: Marketing strategies and tactics Customer fulfillment, training, and support Getting your product out the door Using consultants effectively Understanding cash flow Includes a guide to other business resources.

Text Processing in Python

The most common use for client-server technology is to combine the graphical display capabilities of a desktop PC with the database and number-crunching power of a large central system. But peer-to-peer programs can run equally well.

Programming with Curses

This book shows you how to use two Unix utilities, lex andyacc, in program development. These tools help programmers build compilers and interpreters, but they also have a wider range of applications.The second edition contains completely revised tutorial sections for novice users and reference sections for advanced users. This edition is twice the size of the first and has an expanded index.The following material has been added: Each utility is explained in a chapter that covers basic usage and simple, stand-alone applications How to implement a full SQL grammar, with full sample code Major MS-DOS and Unix versions of lex and yacc are explored in depth, including AT&T lex and yacc, Berkeley yacc, Berkeley/GNU Flex, GNU Bison, MKS lex andyacc, and Abraxas PCYACC

LPI Linux Certification in a Nutshell

The Guide to Writing DCE Applications is a hands-on programming guide to OSF's

Distributed Computing Environment (DCE) for first-time DCE application programmers. This book is designed to help new DCE users make the transition from conventional nondistributed applications programming to distributed DCE programming. Topics include the IDL and ACF files, essential RPC calls, binding methods and the name service, server initialization, memory management, object UUIDs, authentication and authorization (basic security), and other selected advanced topics. Several small, practical programming examples are included. The second edition of this book extends the step-by-step treatment to two advanced topics: object UUIDs and security. Object UUIDs let the client select resources on the server side. Security enforces access on the basis of who the client or server is. This book does not discuss Access Control Lists, a more advanced area of security. We believe the programmer writing her first DCE application will find this book the perfect resource. The Guide to Writing DCE Applications is designed to get the programmer up and running with working examples of client/server applications and to provide a comprehensive model for the development of distributed applications. Contents include: Overview of a distributed application Using a DCE RPC interface (IDL and ACF files) Developing clients Using pointers and arrays Developing a server Using a name service Resource selection through object UUIDs Security (authentication/authorization) for both client and server Using a context handle to maintain server state Using pipes for large quantities of data Quick references Complete code for seven applications, plus Makefiles and directions for building and running

Bison

Understanding windows; Terminal independence; The curses library; Sample program; Quick reference.

UNIX for FORTRAN Programmers

A guide to color management using Adobe Photoshop.