原版风暴 • 深入 C++ 系列

PEARSON
Addison Wesley

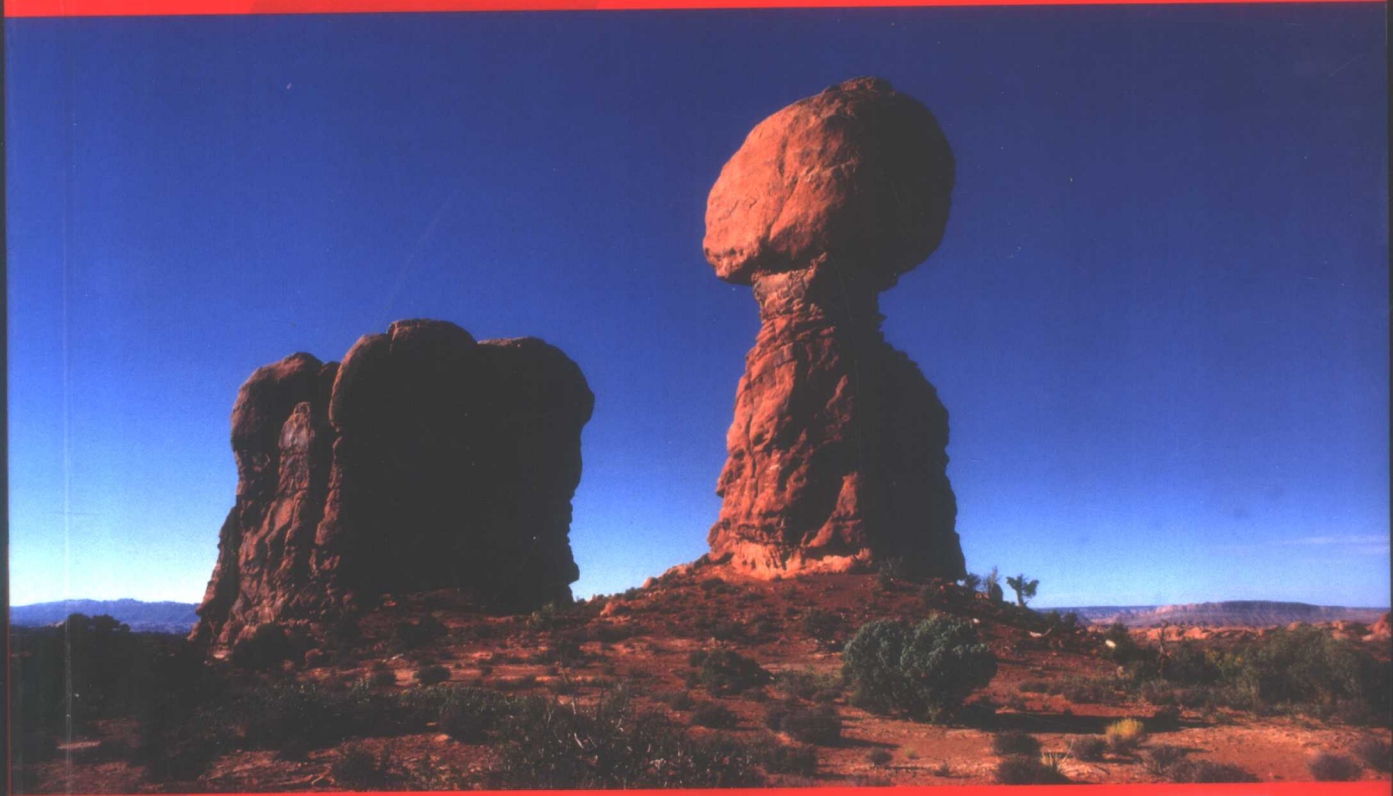
Effective C++ Second Edition

50 Specific Ways to Improve Your Programs and Designs

Effective C++

（第二版·影印版）

[美] Scott Meyers 著



Pearson Education

中国电力出版社
www.infopower.com.cn

Effective C++

Second Edition

原版风暴 • 深入 C++ 系列

Effective C++ Second Edition

50 Specific Ways to Improve Your Programs and Designs

Effective C++

（第二版·影印版）

[美] Scott Meyers 著

中国电力出版社

Effective C++ : 50 specific ways to improve your programs and designs,2nd ed.
（ISBN 0-201-92488-9）
Meyers，Scott

北京市版权局著作合同登记号：图字：01-2003-1013

图书在版编目（CIP）数据

Effective C++ /（美）迈耶斯著. —影印本. —北京：中国电力出版社，2003
（原版风暴·深入 C++系列）
ISBN 7-5083-1498-0
Ⅰ.E... Ⅱ.迈... Ⅲ.C 语言—程序设计—英文 Ⅳ.TP312
中国版本图书馆 CIP 数据核字（2003）第 023077 号

责任编辑：宋宏
丛 书 名：原版风暴·深入C++系列
书　　名：Effective C++（影印版）
编　　著：（美）Scott Meyers
出 版 者：中国电力出版社
地址：北京市三里河路6号　邮政编码：100044
电话：（010）88515918　传真：（010）88423191
印　　刷：北京地矿印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印　　张：**17.5
书　　号：ISBN 7-5083-1498-0
版　　次：2003年6月北京第一版
印　　次：2003年6月第一次印刷
定　　价：40.00 元

For Nancy,
without whom nothing
would be much worth doing.

Wisdom and beauty form a very rare combination.

– Petronius Arbiter
Satyricon, XCIV

引　介

我最早是从 2001 年初《程序员》杂志上侯捷先生的一篇书评文章中听说《Effective C++》这本书的。不久之后，在侯先生的帮助下，我非常幸运地成为国内较早阅读此书的 C++学习者之一。我还清楚地记得，侯先生在给我的电子邮件里让我精心阅读此书，说这本书"将会大幅度提高你的 C++功力"。

其时我学习 C++已经有四年，也曾经反复阅读和学习了好几本颇为有名的 C++教程，对自己的 C++能力不敢说自傲，也略有几分信心。然而初读此书时，心情确实极为震骇。读 Scott Meyers 的这本书就好像在与一名高超的武术教师拆招对战，他招招指向我的要害，而我的种种算计和辩解却似乎早在他预料之中，他能够轻易击溃我建立在无知基础上的所谓信心，之后又辛辣而畅快淋漓地把真本领和盘托出。读这本书的感觉，就像是吃麻辣火锅，既被刺激得大汗淋漓，又不由得从心里直呼过瘾。

从那时起到现在，又过了两年多。这段时间里，不仅这本书的中文版早已经上市，其姊妹篇《More Effective C++》以及其他大批经典 C++书籍也已经纷纷出版。然而今天再来读这本书，我仍然会有一种异样的感觉。虽然里面的内容似乎早已经熟悉，但是那流畅的文字，似乎能猜透读者内心的分析，直击要害的果断评论，仍然让我时时汗颜。

我想 Scott Meyers 在这本《Effective C++》中间所表现出来的东西，实际上还没有被我们所充分认识。我认为，实际上在这本书中，Scott Meyers 创造了一种新的技术文体，一种不同寻常的技术传播方式，他不是在干巴巴地说教，也不是故做高雅地写抒情散文，而是紧盯着你的眼睛，洞悉你的心思，与你进行激烈的技术对话。这种方式在知识传授上的有效性，以及给读者在情绪上所带来的技术亢奋，都是别的方式所无法企及的。而这一切，又跟 Meyers 多年的交互式培训授课经验密切相关。读这本书的时候，我经常在想，我们是不是也应该对国内目前的技术培训和传播方式做一些反思了呢？

非常高兴这本书的英文版即将与我们中国程序员见面。事实上，这本书的侯捷译本是国内少有的高质量中文技术译作。而此次英文版又面市，可谓双剑合璧，对于 C++的学习者和使用者有很大的帮助。作为一个普通的程序员，我为此颇感欣然，故此推介，愿与广大 C++学习者共勉。

孟　岩

2003 年 4 月于北京

Preface

This book is a direct outgrowth of my experiences teaching C++ to professional programmers. I've found that most students, after a week of intensive instruction, feel comfortable with the basic constructs of the language, but they tend to be less sanguine about their ability to put the constructs together in an effective manner. Thus began my attempt to formulate short, specific, easy-to-remember guidelines for effective software development in C++: a summary of the things experienced C++ programmers almost always do or almost always avoid doing.

I was originally interested in rules that could be enforced by some kind of `lint`-like program. To that end, I led research into the development of tools to examine C++ source code for violations of user-specified conditions.† Unfortunately, the research ended before a complete prototype could be developed. Fortunately, several commercial C++-checking products are now available.

Though my initial interest was in programming rules that could be automatically enforced, I soon realized the limitations of that approach. The majority of guidelines used by good C++ programmers are too difficult to formalize or have too many important exceptions to be blindly enforced by a program. I was thus led to the notion of something less precise than a computer program, but still more focused and to-the-point than a general C++ textbook. The result you now hold in your hands: a book containing 50 specific suggestions on how to improve your C++ programs and designs.

In this book, you'll find advice on what you should do, and why, and what you should not do, and why not. Fundamentally, of course, the whys are more important than the whats, but it's a lot more conve-

---

† You can find an overview of the research at the *Effective C++* World Wide Web site: `http://www.awl.com/cp/ec++.html`.

nient to refer to a list of guidelines than to memorize a textbook or two.

Unlike most books on C++, my presentation here is not organized around particular language features. That is, I don't talk about constructors in one place, about virtual functions in another, about inheritance in a third, etc. Instead, each discussion in the book is tailored to the guideline it accompanies, and my coverage of the various aspects of a particular language feature may be dispersed throughout the book.

The advantage of this approach is that it better reflects the complexity of the software systems for which C++ is often chosen, systems in which understanding individual language features is not enough. For example, experienced C++ developers know that understanding inline functions and understanding virtual destructors does *not* necessarily mean you understand inline virtual destructors. Such battle-scarred developers recognize that comprehending the *interactions* between the features in C++ is of the greatest possible importance in using the language effectively. The organization of this book reflects that fundamental truth.

The disadvantage of this design is that you may have to look in more than one place to find everything I have to say about a particular C++ construct. To minimize the inconvenience of this approach, I have sprinkled cross-references liberally throughout the text, and a comprehensive index is provided at the end of the book.

In preparing this second edition, my ambition to improve the book has been tempered by fear. Tens of thousands of programmers embraced the first edition of *Effective C++*, and I didn't want to destroy whatever characteristics attracted them to it. However, in the six years since I wrote the book, C++ has changed, the C++ library has changed (see Item 49), my understanding of C++ has changed, and accepted usage of C++ has changed. That's a lot of change, and it was important to me that the technical material in *Effective C++* be revised to reflect those changes. I'd done what I could by updating individual pages between printings, but books and software are frighteningly similar — there comes a time when localized enhancements fail to suffice, and the only recourse is a system-wide rewrite. This book is the result of that rewrite: *Effective C++*, Version 2.0.

Those familiar with the first edition may be interested to know that every Item in the book has been reworked. I believe the overall structure of the book remains sound, however, so little there has changed. Of the 50 original Items, I retained 48, though I tinkered with the word-

ing of a few Item titles (in addition to revising the accompanying discussions). The retired Items (i.e., those replaced with completely new material) are numbers 32 and 49, though much of the information that used to be in Item 32 somehow found its way into the revamped Item 1. I swapped the order of Items 41 and 42, because that made it easier to present the revised material they contain. Finally, I reversed the direction of my inheritance arrows. They now follow the almost-universal convention of pointing from derived classes to base classes. This is the same convention I followed in my 1996 book, *More Effective C++*, an overview of which you can find on pages 237–238 of this volume.

The set of guidelines in this book is far from exhaustive, but coming up with good rules — ones that are applicable to almost all applications almost all the time — is harder than it looks. Perhaps you know of additional guidelines, of more ways in which to program effectively in C++. If so, I would be delighted to hear about them.

On the other hand, you may feel that some of the Items in this book are inappropriate as general advice; that there is a better way to accomplish a task examined in the book; or that one or more of the technical discussions is unclear, incomplete, or misleading. I encourage you to let me know about these things, too.

Donald Knuth has a long history of offering a small reward to people who notify him of errors in his books. The quest for a perfect book is laudable in any case, but in view of the number of bug-ridden C++ books that have been rushed to market, I feel especially strongly compelled to follow Knuth's example. Therefore, for each error in this book that is reported to me — be it technical, grammatical, typographical, or otherwise — I will, in future printings, gladly add to the acknowledgments the name of the first person to bring that error to my attention.

Send your suggested guidelines, your comments, your criticisms, and — sigh — your bug reports to:

Scott Meyers
c/o Publisher, Corporate and Professional Publishing
Addison Wesley Longman, Inc.
1 Jacob Way
Reading, MA 01867
U. S. A.

Alternatively, you may send electronic mail to `ec++@awl.com`.

I maintain a list of changes to this book since its first printing, including bug-fixes, clarifications, and technical updates. This list is available at the *Effective* C++ World Wide Web site, `http://www.awl.com/cp/ec++.html`. If you would like a copy of this list, but you lack access to the World Wide Web, please send a request to one of the addresses above, and I will see that the list is sent to you.

If you'd like to be notified when I make changes to this book, consider joining my mailing list. For details, consult `http://www.aristeia.com/MailingList/index.html`.

SCOTT DOUGLAS MEYERS

STAFFORD, OREGON
JULY 1997

Acknowledgments

Some three decades have elapsed since Kathy Reed taught me what a computer was and how to program one, so I suppose this is really all her fault. In 1989, Donald French asked me to develop C++ training materials for the Institute for Advanced Professional Studies, so perhaps he should shoulder some blame. The students in my class at Stratus Computer the week of June 3, 1991, were not the first to suggest I write a book summarizing the pearls of alleged wisdom that tumble forth when I teach, but they were the ones who finally convinced me to do it, so they bear some of the responsibility. I'm grateful to them all.

Many of the Items and examples in this book have no particular source, at least not one I can remember. Instead, they grew out of a combination of my own experiences using and teaching C++, those of my colleagues, and opinions expressed by contributors to the Usenet C++ newsgroups. Many examples that are now standard in the C++ teaching community — notably strings — can be traced back to the initial edition of Bjarne Stroustrup's *The C++ Programming Language* (Addison-Wesley, 1986). Several of the Items found here (e.g., Item 17) can also be found in that seminal work.

Item 8 includes an implementation idea from Steve Clamage's May 1993 *C++ Report* article, "Implementing `new` and `delete`." Item 9 was motivated by commentary in *The Annotated C++ Reference Manual* (see Item 50), and Items 10 and 13 were suggested by John Shewchuk. The implementation of `operator new` in Item 10 is based on presentations in the second edition of Stroustrup's *The C++ Programming Language* (Addison-Wesley, 1991) and Jim Coplien's *Advanced C++: Programming Styles and Idioms* (Addison-Wesley, 1992). Dietmar Kühl pointed out the undefined behavior I describe in Item 14. Doug Lea provided the aliasing examples at the end of Item 17. The idea of using `0L` for `NULL` in Item 25 came from Jack Reeves's March 1996 *C++ Report* ar-

ticle, "Coping with Exceptions." Several members of various Usenet C++ newsgroups helped refine that Item's class for implementing NULL-based pointer conversions via member templates. A newsgroup posting by Steve Clamage tempered my enthusiasm for references to functions in Item 28. Item 33 incorporates observations from Tom Cargill's *C++ Programming Style* (Addison-Wesley, 1992), Martin Carroll's and Margaret Ellis's *Designing and Coding Reusable C++* (Addison-Wesley, 1995), *Taligent's Guide to Designing Programs* (Addison-Wesley, 1994), Rob Murray's *C++ Strategies and Tactics* (Addison-Wesley, 1993), as well as information from publications and newsgroup postings by Steve Clamage. The material in Item 34 benefited from my discussions with John Lakos and from reading his book, *Large-Scale C++ Software Design* (Addison-Wesley, 1996). The envelope/letter terminology in that Item comes from Jim Coplien's *Advanced C++: Programming Styles and Idioms*; John Carolan coined the delightful term, "Cheshire Cat class." The rectangle/square example of Item 35 is taken from Robert Martin's March 1996 *C++ Report* column, "The Liskov Substitution Principle." A long-ago `comp.lang.c++` posting by Mark Linton set me straight in my thinking about grasshoppers and crickets in Item 43. My traits examples in Item 49 are taken from Nathan Myers's June 1995 *C++ Report* article, "A New and Useful Template Technique: Traits," and Pete Becker's "C/C++ Q&A" column in the November 1996 *C/C++ User's Journal*; my summary of C++'s internationalization support is based on a pre-publication book draft by Klaus Kreft and Angelika Langer. Of course, "Hello world" comes from *The C Programming Language* by Brian Kernighan and Dennis Ritchie (Prentice-Hall, initially published in 1978).

Many readers of the first edition sent suggestions I was unable to incorporate in that version of the book, but that I've adopted in one form or another for this new edition. Others took advantage of Usenet C++ newsgroups to post insightful remarks about the material in the book. I'm grateful to each of the following individuals, and I've noted where I took advantage of their ideas: Mike Kaelbling and Julio Kuplinsky (Introduction); a person my notes identify only as "a guy at Claris"† (Item 5); Joel Regen and Chris Treichel (Item 7); Tom Cargill, Larry Gajdos, Doug Morgan, and Uwe Steinmüller (Item 10); Roger Scott and Steve Burkett (Item 12); David Papurt (Item 13); Alexander Gootman (Item 14); David Bern (Item 16); Tom Cargill, Tom Chappell, Dan Fran-

---

† Note to this guy: I was at Claris the week of November 15, 1993. Contact me and identify yourself as the one who pointed out the importance of specifying which form of `delete` to use with a `typedef`, and I'll happily give you proper credit in these acknowledgments. I'll even throw in a little something (*very* little — don't get excited) to help compensate for my pathetic failure to know who you are.

klin, and Jerry Liebelson (Item 17); John "Eljay" Love-Jensen (Item 19); Eric Nagler (Item 22); Roger Eastman, Doug Moore, and Aaron Naiman (Item 23); Dat Thuc Nguyen (Item 25); Tony Hansen, Natraj Kini, and Roger Scott (Item 33); John Harrington, Read Fleming, and David Smallberg (Item 34); Johan Bengtsson (Item 36); Rene Rodoni (Item 39); Paul Blankenbaker and Mark Somer (Item 40); Tom Cargill and John Lakos (Item 41); Frieder Knauss and Roger Scott (Item 42); David Braunegg, Steve Clamage, and Dawn Koffman (Item 45); Tom Cargill (Item 46); Wesley Munsil (Item 47); Randy Mangoba (most class definitions); and John "Eljay" Love-Jensen (many places where I use type double).

Partial and/or complete drafts of the manuscript for the first edition were reviewed by Tom Cargill, Glenn Carroll, Tony Davis, Brian Kernighan, Jak Kirman, Doug Lea, Moises Lejter, Eugene Santos, Jr., John Shewchuk, John Stasko, Bjarne Stroustrup, Barbara Tilly, and Nancy L. Urbano. In addition, I received suggestions for improvements that I was able to incorporate in later printings from the following alert readers, whom I've listed in the order in which I received their reports: Nancy L. Urbano, Chris Treichel, David Corbin, Paul Gibson, Steve Vinoski, Tom Cargill, Neil Rhodes, David Bern, Russ Williams, Robert Brazile, Doug Morgan, Uwe Steinmüller, Mark Somer, Doug Moore, David Smallberg, Seth Meltzer, Oleg Shteynbuk, David Papurt, Tony Hansen, Peter McCluskey, Stefan Kuhlins, David Braunegg, Paul Chisholm, Adam Zell, Clovis Tondo, Mike Kaelbling, Natraj Kini, Lars Nyman, Greg Lutz, Tim Johnson, John Lakos, Roger Scott, Scott Frohman, Alan Rooks, Robert Poor, Eric Nagler, Antoine Trux, Cade Roux, Chandrika Gokul, Randy Mangoba, and Glenn Teitelbaum. Each of these people was instrumental in improving the book you now hold.

Drafts of the second edition were reviewed by Derek Bosch, Tim Johnson, Brian Kernighan, Junichi Kimura, Scott Lewandowski, Laura Michaels, David Smallberg, Clovis Tondo, Chris Van Wyk, and Oleg Zabluda. I am grateful to all these people, but especially to Tim Johnson, whose detailed review influenced the final manuscript in dozens of ways. I am also grateful to Jill Huchital and Steve Reiss for their assistance in finding good reviewers, a task of crucial importance and increasing difficulty. Dawn Koffman and David Smallberg suggested improvements to the C++ training materials derived from my books, and many of their ideas have found their way into this revision. Finally, I received comments from the following readers of earlier printings of this book, and I've modified this current printing to take their suggestions into account: Daniel Steinberg, Arunprasad Marathe, Doug Stapp, Robert Hall, Cheryl Ferguson, Gary Bartlett, Michael

Tamm, Kendall Beaman, Eric Nagler, Max Hailperin, Joe Gottman, Richard Weeks, Valentin Bonnard, Jun He, Tim King, Don Maier, Ted Hill, Mark Harrison, Michael Rubenstein, Mark Rodgers, David Goh, Brenton Cooper, Andy Thomas-Cramer, Antoine Trux, John Wait, Brian Sharon, Liam Fitzpatrick, Bernd Mohr, Gary Yee, John O'Hanley, Brady Patterson, Christopher Peterson, Feliks Kluzniak, Isi Dunietz, Christopher Creutzi, Ian Cooper, Carl Harris, Mark Stickel, Clay Budin, Panayotis Matsinopoulos, David Smallberg, Herb Sutter, Pajo Misljencevic, Giulio Agostini, Fredrik Blomqvist, and Jimmy Snyder.

Evi Nemeth (with the cooperation of Addison-Wesley, the USENIX Association, and The Internet Engineering Task Force) has agreed to see to it that leftover copies of the first edition are delivered to computer science laboratories at universities in Eastern Europe; these universities would otherwise find it difficult to acquire such books. Evi voluntarily performs this service for several authors and publishers, and I'm happy to be able to help in some small way. If you'd like more information on this program, contact Evi at `evi@cs.colorado.edu`.

Sometimes it seems that the players in publishing change nearly as frequently as the trends in programming, so I'm pleased that my editor, John Wait, my marketing director, Kim Dawley, and my production director, Marty Rabinowitz, continue to play the roles they did in those innocent days of 1991 when I first started this whole authoring thing. Sarah Weaver was my project manager for this book, Rosemary Simpson provided advice on indexing, and Lana Langlois acted as my primary contact and all-around *über*coordinator at Addison-Wesley until she left for greener — or at least different — pastures. I thank them and their colleagues for helping with the thousand tasks that separate simple writing from actual publishing.

Kathy Wright had nothing to do with the book, but she'd like to be acknowledged.

For the first edition, I am grateful for the enthusiastic and unflagging encouragement provided by my wife, Nancy L. Urbano, and by my family and hers. Although writing a book was the last thing I was supposed to be doing, and doing so reduced my free time from merely little to effectively none, they made it clear that the effort was worth it if, in the end, the result was an author in the family.

That author has been in the family six years now, yet Nancy continues to tolerate my hours, put up with my technochatter, and encourage my writing. She also has a knack for knowing *just* the right word when I can't think of it. The Nancyless life is not worth living.

Our dog, Persephone, never lets me confuse my priorities. Deadline or no deadline, the time for a walk is always *now*.

Praise for the first edition of *Effective C++:*

"Meyers's book I must really praise. ... The book also contains an excellent nuts and bolts guide to memory management constructs, and a great explanation of the meaning of the different types of C++ inheritance."
–*New York Computerist*

"This should definitely be read before you start your first real C++ project and reread again as you gain experience."
–comp.lang.c++

"Subtitled '50 Specific Ways to Improve Your Programs and Designs,' the author offers not only explicit rules to follow when writing C++ code, but provides justification and examples illustrating their use."
–*Sun Expert*

"I heartily recommend *Effective C++* to anyone who aspires to mastery of C++ at the intermediate level or above."
–*The C User's Journal*

"This is one of the best books that I've seen for the 'intermediate-level' programmer. It's structured as a series of 'essays' on practical problems that a C++ programmer encounters. ...It is the rare programming book that is both funny and useful."
–comp.lang.c++

"The result is a small book analogous in scope and flavor to another little book, *The Elements of Style* by William Strunk and E.B. White. On my shelf, at least, the two books are not far apart. ...This is a modest little book that sets clear goals and achieves them."
–*C++ Report*

"This book contains practical advice on exploiting C++."
–*DEC Professional*

"Any C++ programmer should definitely not only own but study and put to use this book. This text is easy to use and well cross-referenced and indexed."
–*Computer Language*

"This is a 193-page masterpiece that I read in one sitting. ... I guarantee that some combination of these 50 items will grab and enlighten you, and repay your modest investment. ... This is a well-written, honest book aimed at C++ programmers who are converging toward fluency and effectiveness."
–Stan Kelley-Bootle, *UNIX Review*

Contents