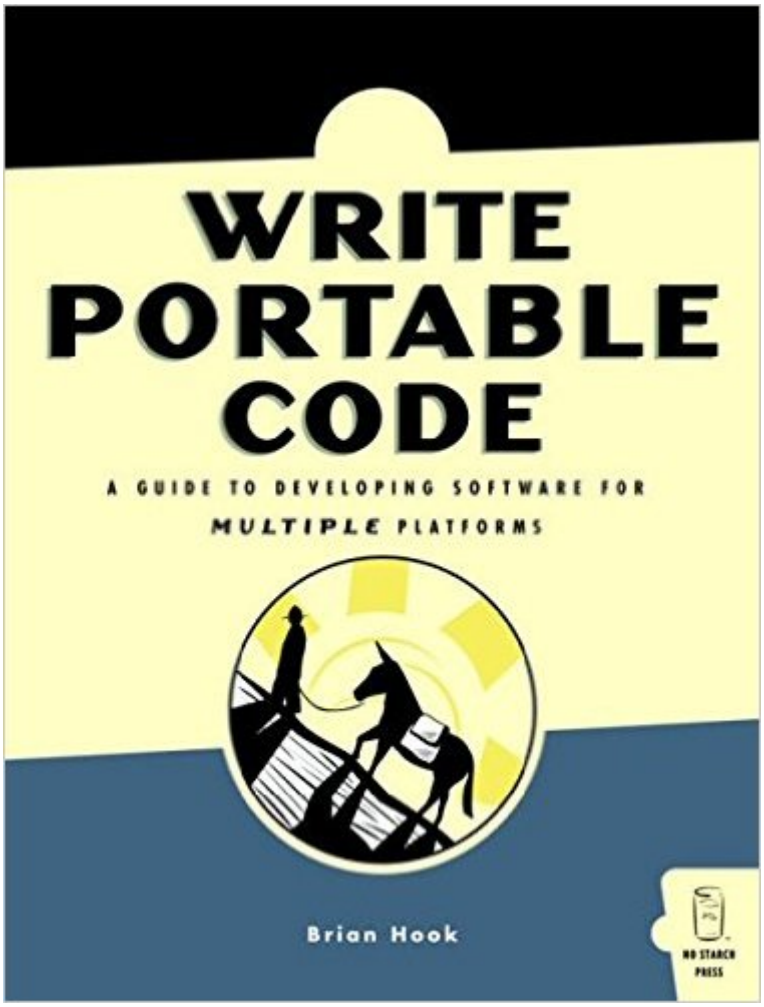
Write Portable Code: An Introduction To Developing Software For Multiple Platforms



Synopsis

Portable software development is writing software that runs on a broad range of computer systems instead of just one (e.g., Windows). Programmers often pick up the idioms, tricks and methodologies for developing cross-platform software through sheer trial and error, as they encounter the same mistakes and patterns of code over time. If you’re an intermediate-to advanced-level programmer who’d rather cut to the chase, Write Portable Code contains the lessons, patterns and knowledge you’ll need for developing cross-platform software.Write Portable Code explains how to: avoid common portability mistakes when starting out a new project, thereby saving time when a port must occurre-factor existing, non-portable code so that it can be easily transplanted to new platformsfind bugs masked by platform specific behaviorsProgrammers who avoid becoming married to a specific development environment or target platform greatly expand the target market for their software products. Whether you design cross-platform software from the ground up or have to move large amounts of code from one platform to another, the information contained in Write Portable Code will help you achieve your goals and grow as a programmer.TOCPrefaceIntroductionChapter 1: Preparing for PortabilityChapter 2: ANSI C/C++Chapter 3: Techniques for PortabilityChapter 4: Editing and Source ControlChapter 5: Processor DifferencesChapter 6: Floating PointChapter 7: PreprocessorChapter 8: Compiler QuirksChapter 9: User InteractionChapter 10: NetworkingChapter 11: Operating SystemsChapter 12: Dynamic LibrariesChapter 13: Security and PermissionsChapter 14: File SystemsChapter 15: Scalability and PortabilityChapter 16: Portability and DataChapter 17: Internationalization and LocalizationChapter 18: Scripting LanguagesChapter 19: Cross-platform Libraries and ToolkitsAppendix A: POSHAppendix B: The Simple Audio LibraryAppendix C: The Rules for PortabilityReferences

Book Information

Paperback: 272 pages
Publisher: No Starch Press; 1 edition (July 15, 2005)
Language: English
ISBN-10: 1593270569
ISBN-13: 978-1593270568
Product Dimensions: 7 x 0.8 x 9.2 inches
Shipping Weight: 1.2 pounds (View shipping rates and policies)
Average Customer Review: 4.1 out of 5 starsÂ Â See all reviewsÂ (13 customer reviews)

Best Sellers Rank: #1,410,995 in Books (See Top 100 in Books) #31 inÂ Books > Computers & Technology > Programming > Cross-platform Development #3813 inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Software Development #8974 inÂ Books > Computers & Technology > Programming > Languages & Tools

Customer Reviews

This book is an excellent read. The material is presented completely, concisely, and in an easy to understand manner. However the title of this book is misleading. It really should be "Write Portable code in C". There is some limited discussion of C++, but mostly to discourage you from using it. All other languages are dismissed out of hand, or just completely ignored. As long as you understand this caveat there is much to learn from this book.This book is really focused on writing software that will run on essentially any platform that has a C compiler, which today is almost all processors. If you need to write software that will run on embedded 16 bit processors as well as the latest 64 bit ones, then you should read this book. However, there are large classes of software that have a more limited notion of portability (such as running on most 32 bit Unix or Windows platforms, or any platform that g++ can target) where Standard C++ or Java are the way to go. Unfortunately the book does not adequately address the tradeoffs, design, and implementation decisions one should make in these cases. In particular, I am puzzled by the total lack of Java solutions.Since the book emphasizes C programming, there is minimal content on GUI programming, Web programming, database programming, and similar areas where C programming is rarely used anymore.

Enter Brian Hook’s Write Portable Code. Portability is a sort of a holy grail for programmers, and there is no lack of knowledge floating around Usenet and Internet. However, Brian Hook tried to recommend a book on the subject to a friend, and didn’t find one. So he set out to write his own.The result is a tome that should reside on every programmers desk sooner or later. The book’s cover pictures a donkey striving uphill, and I can definitely releate to that. Portability is not only a target hard to aquire, but it is also a target roaming about in a minefield. I consider myself a best-practice programmer: stability and portability is important to me. However, while reading this book, I quickly realized that I know nothing. Each and every chapter contained possible gotchas that have eluded me for years. This book does two things: it points out those gotchas, and it gives the reader oodles of clever tricks, background info and solutions. And best of all, those tricks are often of the stablest possible quality.Diving into this book, I had to break up old ideas about what is safe and not. For can you rely on sizeof(int)? Are you sure main() is your entry point? What if you need to move to a

mixed-endian platform? Westwood's assumptions about floating point behaviour made it impossible for Aspyr to implement network play across platforms for Command & Conquer: Generals. Deciding on portability is also a matter of choosing your platforms. Hook points out that you can write a program that runs the same on a clustered super computer as well as a coffee machine, but it doesn't make much sense. Instead, establishing a baseline will go a long way towards keeping your sanity. He does not, however, back down from tricky configurations: there are recommendations and warnings that apply to esoteric platforms, e.g. embedded environments with very limited resources.Many of the examples in the book are drawn from a cross-platform audio library (SAL) that Brian Hook has written. Seeing the concepts in the book in their actual implementation brings a lot to the table - his treatise points out both goals, possible problems, and solutions.Since the book is so C++-centric, I would have valued information on the Standard Template Library (STL). While its name promises a standard, it has a few extensions that might not always be around. Performance characteristics also vary across implementations.Brian Hook has a remarkably clear and readable language. Even though the material in the book is complex, and Hook is writing in a very to-the-point manner, every nuance of the issue gets across. A result of many hours of editing, I'm sure, and those were hours well spent. I finally managed to get the ISO, ANSI, IEEE and many of their different standards sorted out in my head, no small feat for Brian!Achieving portability is a matter of being extremely cautious, and it is surrounded by many misconceptions and myths. In that regard, Provided that Brian is right (which I'm really not qualified to tell), he becomes a sort of myth-buster. However, since he comes from a background with portability engineering at id Software and 3Dfx, he does have the history to back up his tricks.The book is very C/C++-centric, but many of the concepts are directly portable to other languages such as (yes) Java. As for the code samples, they are a bit long-winded at times, but always clear. My only gripe is that some of the Mac source code samples aren't always optimal - they are correct and work, but could sometimes have been accomplished in a less complex way. That is, however, mostly a question of form.

This is the only book I've seen that covers this material in-depth (The Practice of Programming by Kernighan and Pike touches on the basics). If you've never worked on the development of a piece of cross-platform software, this book will help you avoid a lot of the pitfalls that many programmers have overcome through trial and error. Otherwise, there's not much novel material here for you.The first half of this book decribes the method for writing portable code with a good high-level philosophy (introduce abstraction and indirection wherever necessary) as well as delving into a lot of the

specifics of how to accomplish it. All the bases are covered. The second half of the book offers an unfortunately shallow review of the different subsystems that differ between platforms without giving any real options for how to bridge the gaps.Overall, the book is well written and correct. On the downside there's nothing ground-breaking here and the effective length of the book is about half of its actual length given the disappointing second half.

This book is an in-depth discussion of issues involved in getting C/C++ code from one platform to another. Hook dives deep into arcane topics such as processor memory access alignment, floating point operation platform differences, and exception handling. The book's not for the faint-of-heart, and it's rather specific to C/C++; however, readers brave enough to push through the book should get interesting insights regardless of what platform and development environment they're working with.This appears to be a great resource for folks who are actively porting software. It's a very good guidebook of issues to address if you're even thinking about porting. I'd also say it's a good skimming read for most developers if only to get an understanding of some engineering principles to consider when building your systems. You never know when your platform might fold or get deprecated.

This book does a really great job at talking about code portability. It definitely was an eye opener. However, it started loosing me around chapter five so I paused to increase my knowledge of programming and I will return it to later when my programming understanding increases.

International Private Sewage Disposal Code) (International Code Council Series) Embedded Linux Porting on ARM & RFID Implementation Using ARM SoC: Developing a flexible and agile Board Secure Package Linux with multiple applications Microsoft Win32 Programmer's Reference: Introduction Platforms, and Index (Microsoft Windows Programmer's Reference Library) Software Engineering Classics: Software Project Survival Guide/ Debugging the Development Process/ Dynamics of Software Development (Programming/General) Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection: Obfuscation, Watermarking, and Tamperproofing for Software Protection Piano Literature - Book 3: Developing Artist Original Keyboard Classics (The Developing Artist Library) Piano Sonatinas - Book One: Developing Artist Original Keyboard Classics (The Developing Artist) Piano Literature - Book 4: Developing Artist Original Keyboard Classics (The Developing Artist) COM Beyond Microsoft: Designing and Implementing COM Servers on Compaq Platforms (HP Technologies) Laboratory Manual to Accompany Security Strategies in Linux Platforms and Applications (ITT Edition IS3440)