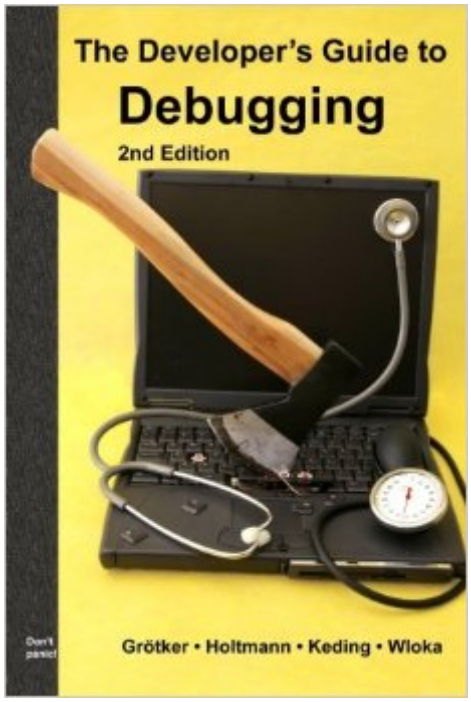
The Developer's Guide To Debugging: 2nd Edition



Synopsis

Software has bugs. Period. That’s true, unfortunately. Even the good old "hello, world" program, known to virtually every C and C++ programmer in the world, can be considered to be buggy. Developing software means having to deal with defects; old ones, new ones, ones you created yourself and those that others brought to life. Software developers debug programs for a living. Hence, good debugging skills are a must-have. That said, I always found it regretable that debugging is hardly taught in engineering schools. Well, it is a tricky subject, and there are no good textbooks. The latter can be helped, I thought. That’s how the idea for this book was born. "The Developer’s Guide to Debugging" is a book for both professional software developers seeking to broaden their skills and students that want to learn the tricks of the trade from the ground up. With small inlined examples and exercises at the end of each chapter it is well suited to accompany a CS course or lecture. At the same time it can be used as a reference used to address problems as the need arises. This book goes beyond the level of simple source code debugging scenarios. In addition, it covers the most frequent real-world problems from the areas of program linking, memory access, parallel processing and performance analysis. The picture is completed by chapters covering static checkers and techniques to write code that leans well towards debugging. While the focus lies on C and C++, the workhorses of the software industry, one can apply most principles described in "The Developer’s Guide to Debugging" to programs written in other languages. The techniques are not restricted to a particular compiler, debugger or operating system. The examples are structured such that they can be reproduced with free open-source software.

Book Information

Paperback: 242 pages

Publisher: CreateSpace Independent Publishing Platform; 2 edition (April 22, 2012)

Language: English

ISBN-10: 1470185520

ISBN-13: 978-1470185527

Product Dimensions: 6 x 0.6 x 9 inches

Shipping Weight: 15 ounces (View shipping rates and policies)

Average Customer Review: 4.7 out of 5 starsÂ Â See all reviewsÂ (6 customer reviews)

Best Sellers Rank: #1,115,606 in Books (See Top 100 in Books) #53 inÂ Books > Computers & Technology > Programming > Languages & Tools > Debugging #226 inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Tools #2990 inÂ Books >

Computers & Technology > Programming > Software Design, Testing & Engineering > Software Development

Customer Reviews

I like this book so much that I've given copies of it to my close friends. The Developer's Guide to Debugging is a fantastic little book. This book focuses on the general topic of debugging C and C++ code; however, much of what is said can be useful for other programming languages (e.g. ObjectiveC or C#). Although others have written books on debugging, this book really got to the heart of the matter for me. The focus of the book is software development, but since modern digital design is really software design also, I feel this book should prove equally useful to those doing hardware design also. Of course my specialty, Electronic System-Level design using SystemC, fits perfectly.One of the things I like is that the book is not overly long, and each section has a nice summary of key concepts at the end. I also like that it covers topics for debugging code without debug information and provides strategies for trying to finding hard to repeat bugs. They also point out how debug tools can affect the bug, which brings up the "Heisen bug".Chapter 2, A Systematic Approach to Debugging is the most important chapter of the book. If you don't read anything else, read this chapter...twice. Viewing debug as a process is very important, and I think any engineer, whether hardware or software, will benefit from this insight. We apply rigorous processes to most everything we do in engineering until we get to this. I cannot recall the number of times I've seen engineers pull up a waveform or dive into GDB before they've really considered where the issues are. What usually follows are hours of wasted forays until they stumble on the problem. Follow the systematic approach shown in this book and you will get to the root of the problem much quicker. This chapter covers both strategy and provides a classification system for bugs.Subsequent chapters provide insights into specific areas and provide valuable tips and approaches to recognizing and solving problems in this area. After reading Chapter 2, you don't have to read the book sequentially, but can go directly to any area. I recommend reading chapter 11, which will help you to write code that is easier to debug. The table of contents is a great way to look at this book. In the following, I have added my own comments following a hypen (-) after the chapter titles. 1. You Write Software You Have Bugs - even Hello World has bugs 2. A Systematic Approach to Debugging - Golden Rules and a must read for all engineers 3. Getting to the Root - Source Code Debuggers 4. Fixing Memory Problems - finding memory leaks and bad pointer problems 5. Profiling Memory Use 6. Solving Performance Problems 7. Debugging Parallel Programs - a topic not often dealt with in other books 8. Finding Environment and Compiler Problems 9. Dealing with Linking

Problems 10. Advanced Debugging 11. Writing Debuggable Code - this is invaluable and should be required reading 12. How Static Checking Can Help - finding bugs before you execute a line of code! 13. SummaryA. Debugger Commands - a short list of key GDB and Visual Studio commandsB. Access to tools - an excellent list of tools you might not be familiar withBecause debugging is an ageless topic and because this book looks beyond specific tools, I feel this book will continue to be useful for many years to come.In summary, I highly recommend this book for software developers, verification engineers and system-level designers using SystemC (or any standard programming language). RTL designers might not benefit quite as much, but chapter 2 is worth the read.

I like the book because most of my computer courses taught very little about debugging and how to use a development environment like Microsoft Visual Studion. This book does an excellent job at explaining debugging which is essential to doing a good job at programming.

I've been programming as part of my job for many years and this book expanded my horizons, must read for any systems programmer.Kindle NOTE:[soapbox] Kindle version is OK for this book if you are reading cover to cover but it shows the usual artifacts of an automatic conversion without a human editor reviewing the output and fixing things like chapter headings typeset in the same font/size as the body text. IMO kindle is generally worthless for textbooks, technical manuals and references because of its naive implementation and lack of features including syntax high-lighting, linking words to glossaries, complete navigational interface, scrolling text one line at a time (to align an image with its caption on the same page for instance), inline mathematics that respect background colors and alignments, and a full text / full notes boolean search that organizes hits for individual review. [/soapbox]Languages: C/C++ centric (all examples), most of this book applies to ALL imperative languages.Operating System: It is UNIX centric but includes information for MS Visual Studio.Full Disclosure:I don't even know what VS looks like, The only time I ever use windows is fixing it for a friend, so my opinion could be worthless on Visual Studio, but the information is there. I use Macs but don't program with XCode so I can't give you any info on that specifically at all.In my experience Debugging is both an intuitive art (gained by years of experience working on real machines with real code) and a very demanding science (making observations, taking notes, well-formed hypotheses, careful testing one step at a time) My favorite quip is in section 2.1 where it mentions a problem solution method suggested by R. Feynman: "Write down the problem, think very hard, write down the answer." which is of course an "always true" statement.

What this book does is help you to understand how to identify the problem so you can write it down (understand it) and then expands on the "think very hard" clause and makes numerous suggestions of how to go about that (solve it). This leads us to another statement of Feynman: ’The key to solving any problem is in looking at the problem in such a way that the solution becomes obvious.’ If you get the depth of that statement, let me say: this book is that good.ALSO:Gives a tutorial on GDB using a subset of commands to get you started with GDB - This tutorial assumes you are learning GDB - not basics of debugging, machine organization and memory layout.Includes an extensive listing of up-to-date development tools, build tools, and testing tools.Gives several insights on debugging library code. (The part I needed most! - very good stuff)Up-to-date Bibliography references as late as 2009, all refs are in 21st century.Includes a xref between basic GDB commands and Visual Studio debugging commands. (Appendix A)This book has increased my skill level and enhanced my understanding of debugging - excellent work by T. GrÃ¶tker.

This book explained a number of debugging techniques, but most importantly it explained when and why to use each one.The instructions were sufficient for me to get started, without too much detail to slog through.

I thought this was a good overall book on debugging. The kindle version has some quirks with spacing but is okay

Great!

ABAP: Part1: Conceptual Design, Development, Debugging CICS/VS: A guide to application debugging (The QED IBM mainframe series) Inside the Microsoft Build Engine: Using MSBuild and Team Foundation Build (2nd Edition) (Developer Reference) Microsoft .NET - Architecting Applications for the Enterprise (2nd Edition) (Developer Reference) Tabular Modeling in Microsoft SQL Server Analysis Services (2nd Edition) (Developer Reference) VBA Developer's Handbook, 2nd Edition Valgrind 3.3 - Advanced Debugging and Profiling for Gnu/Linux Applications Unix System V: Understanding Elf Object Files and Debugging Tools (Programmer Collection)