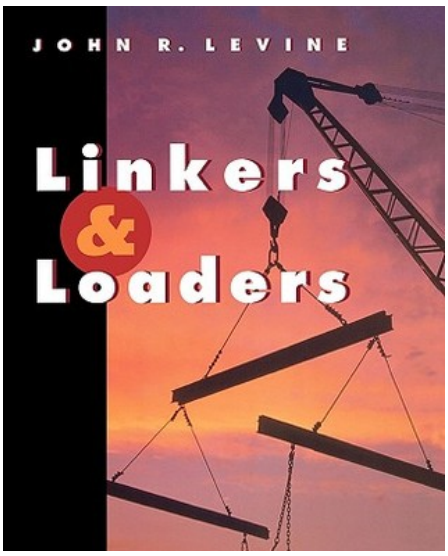

Linkers and Loaders

John R. Levine

Linkers and Loaders

John R. Levine

Linkers and Loaders John R. Levine
Whatever your programming language, whatever your platform, you probably tap into linker and loader functions all the time. But do you know how to use them to their greatest possible advantage? Only now, with the publication of **Linkers & Loaders**, is there an authoritative book devoted entirely to these deep-seated compile-time and run-time processes.
The book begins with a detailed and comparative account of linking and loading that illustrates the differences among various compilers and operating systems. On top of this foundation, the author presents clear practical advice to help you create faster, cleaner code. You'll learn to avoid the pitfalls associated with Windows DLLs, take advantage of the space-saving, performance-improving techniques supported by many modern linkers, make the best use of the UNIX ELF library scheme, and much more. If you're serious about programming, you'll devour this unique guide to one of the field's least understood topics. **Linkers & Loaders** is also an ideal supplementary text for compiler and operating systems courses.

*Includes a linker construction project written in Perl, with project files available for download. *Covers dynamic linking in Windows, UNIX, Linux, BeOS, and other operating systems.
*Explains the Java linking model and how it figures in network applets and extensible Java code.
*Helps you write more elegant and effective code, and build applications that compile, load, and run more efficiently.

Linkers and Loaders Details

Date	:	Published October 25th 1999 by Morgan Kaufmann Publishers (first published October 11th 1999)
ISBN	:	9781558604964
Author	:	John R. Levine
Format	:	Paperback 256 pages
Genre	:	Computer Science, Programming, Science, Technology, Software, Reference

From Reader Review Linkers and Loaders for online ebook

Jason Copenhaver says

The book is certainly dated. But still very useful and a great introduction to linkers and loaders. I wish I had read this book 10 years ago when I was still working with a custom loader implementation.

Jose says

I

Jan Kroken says

Given that it is now around 20 years old, and contains a lot of historical references, the relevancy of the content is so-so. Still a great book on the subject.

Mike says

I tried this out because of Greg Wilson's recommendation at http://www.third-bit.com/reading.html at a point where I was wrestling with some shared library implementation questions.

The book was a little too specific for me and I couldn't get much out of it. My takeaway is that the basic principles of linkers and loaders are straightforward enough but that every single specific implementation is a collection of nasty hacks influenced by the operating system and hardware.

I came away thinking that there *must* be a better set of references online by now; if I had to guess I would start off at Wikipedia.

Carlos O'Donell says

This is an *excellent* introduction into the requirements of both static linkers and dyanmic linkers (loaders). It is a must read for anyone who is thinking about hacking on a linker or loader e.g. the GNU linker or the dynamic loader in the GNU C library.

David Lindelof says

You may have written hundreds, maybe thousands of programs, but if you are like most programmers then everything that happens after the compilation is kind of mysterious. Why does the compiler have to create object files? What are they? What is this so-called linker who combines those files into a library, or an

executable? What's its purpose? John Levine's book answers those questions, and more.

Item 53 in 97 Things Every Programmer Should Know: Collective Wisdom from the Experts is "The Linker Is not a Magical Program", and this book goes a long way towards taking that magic away. It carefully explains step by step what happens from the moment the code is compiled until it actually runs on the machine; and what's more important, it makes it very clear *why* things are as they are today.

I was recommended this book in a reply to a Stackoverflow question, and I am not disappointed. The book goes occasionally perhaps a little bit too much into technical details, which I felt could be safely skipped. Perhaps a case study, i.e. going through every single step towards running a complete program, would have been useful, instead of exposing how different systems solve the different steps one by one.

Until I read this book I simply did not understand how a program actually ran on my computer. A few details are still a bit fuzzy, but now I feel much better equipped for dealing with obscure linker errors or custom linker scripts. Highly recommended for any programmer who wants to get to the bottom of things.

Steve says

Definitely worth reading if you write native code for a living (or hobby).

Published in 2000, it covers Linux (ELF and a.out), Windows (COFF) and a huge mess of older obsure object formats and linking systems. Pretty much the only book of it's kind and a great primer if you're interested in writing your own linker or loader for an OS or just need to better understand native executables.

It sometimes got lost in the ghosts of systems past like the IBM 360 object format, DOS .OBJ files (Intel OMF), real mode (ugh). My only semi-complaint is that it spent a lot of pages talking about real mode but being published in 2000, we were less than a decade removed from that fever dream of backwards compatibility.

Joshua Goller says

This is a hard book for me to rate. I read it because my boss/Jedi master at work said that he wanted me to write a PE loader, and I realized that I had no idea how to do that without using the Windows API. On one hand, I found this book to be really hard to read, but at a second glance, I can't really say there is anything wrong with it except for a two points:

1. It's 15 years old. This is a pretty big deal to me because it's hard to tell what's still useful and what has been deprecated. There's ample discussion of Unix System V, for instance, and some now-mostly-deprecated file formats, like a.out.

2. It feels sort of haphazardly organized. Some of the sections have odd or off-topic subsections, and I think it could be better organized.

If you're thinking of reading this book, my advice is this: do it, but do it lightly. I didn't know what linkers or loaders do going into this book, and while I am still confused about some aspects of them, I feel like I have a better understanding of what happens between object code and executable than I did before. However, while

much of the material is still relevant today (the author does discuss PE and ELF pretty extensively), there is still much ground to cover since its publication.

--Dependencies--
The ELF and PE specifications
C or C++ programming (K&R is probably enough)
x86 assembly (see my other recommendations for this)
Some Wikipedia-ing on memory management and processes on UNIX-like or Windows systems

---

Michael Pankov says

You barely can understand linkers by this book since it falls into implementation details of each specific OS and CPU too early. In the end, the book presents recollection of dirty hacks from existing (and dated) implementation, however with explanations why there're these hacks in the first place. But I'd rather have a book on design of new linkers and loaders, or at least in-depth analysis of at least one loader. I understand new loaders are very rarely designed, but anyway, this book is of no help when they do.

Suvaditya says

Excellent Book !!
