Checking C Programs with Lint by Ian F. Darwin. O'Reilly 1998 (ISBN 0-937175-30-7)

Reviewed by Michael Still <mikal@stillhq.com>

I have a confession to make. When I ordered by review copy of this book I didn't check the publish date — I just assumed it would be relatively recent. Unknown to me, I had just ordered a Unix classic.

My motivation in reviewing this book was a little selfish. I write a lot of C code, and always walk away wondering if smarter people than myself are going to pick holes through it at a later date. I see linting my code as a way of avoid embarrassment later. *Checking C Programs with Lint* discusses how to drive lint, some of the more common error messages, and how to fix them. It also discusses some addin programs which can help lint do it's job.

It also did more than that for me. Any book that in explaining how to use a tool like lint feels that it needs to explain the history of Unix, including diagrams, is trying to tell me something. Lint seems to have a chequered past, with some releases of the "Research" Unix including useful features, and the Berkley people having different command line options from the USG people [1].

There are some other catches, apart from ancient history, in working with a book which is so old. Firstly, I tried to find a lint package for my trusty Debian laptop. There isn't one that I can find — the closest I could get was a lint-like package called splint. A quick Google confirmed that there didn't seem to be any other freely available lint implementations for C code floating around.

"Splint is a tool for statically checking C programs for security vulnerabilities and coding mistakes. With minimal effort, Splint can be used as a better lint. If additional effort is invested adding annotations to programs, Splint can perform stronger checking than can be done by any standard lint." -- http://www.splint.org

So, I'm stuck with using splint to do my linting. Unfortunately, splint implements different command line options.

So, what parts of *Checking C Programs with Lint* are relevant to modern programmers? Well, the historical aspects are certainly interesting, but not going to make me a better programmer. The commentary of how VAXen have different sizes for basic data types certainly made me think about portability, but sizeof() is a pretty standard thing to use these days. The discussion of command line arguements and how they are implemented across a variety of Unix vendors, isn't really going to help me. The examples of common lint error messages and how to fix them is certainly worth thinking about, but don't map to the messages produced by splint.

This review is a little sad in a way. *Checking C Programs with Lint* is a good read, and very well written. It's one of those books I found engaging enough to read from cover to cover in a very small amount of time. However, it's time as a programmer's reference may well have passed. O'Reilly, bring on an update!

Notes

1. Please note, I have no idea who holds the moral high ground on these issues, but I am confident many people will write to tell me. Please just bear in mind that when this book was published, I was in year five.