Pearson

异步图书
www.epubit.com.cn

“笨办法”学C语言

Learn C the HARD WAY

Practical Exercises on the Computational Subjects You Keep Avoiding (Like C)

[美] 泽德 A. 肖（Zed A. Shaw） 著
王巍巍 译

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

Pearson

“笨办法”学C语言

Learn C the HARD WAY

Practical Exercises on the Computational Subjects You Keep Avoiding (Like C)

[美] 泽德 A. 肖（Zed A. Shaw）著
王巍巍 译

人民邮电出版社
北京

图书在版编目（CIP）数据

“笨办法”学C语言 / (美) 泽德·A. 肖 (Zed A. Shaw) 著 ; 王巍巍译. -- 北京 : 人民邮电出版社, 2018.4
书名原文: Learn C the Hard Way: Practical Exercises on the Computational Subjects You Keep Avoiding (Like C)
ISBN 978-7-115-47730-9

Ⅰ. ①笨… Ⅱ. ①泽… ②王… Ⅲ. ①C语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2018)第016523号

内容提要

这本书的目标是让读者掌握足够的 C 语言技能，从而可以自己用 C 语言编写程序或者修改别人的 C 语言代码，成为一名优秀的程序员。但这并不完全是一本讲 C 语言编程的书，书中还重点介绍防御性编程。本书以习题的方式引导读者一步一步学习编程，结构非常简单，共包括 52 个习题，每一个习题都重点讲解一个重要的主题，多数是以代码开始，然后解释代码的编写，再运行并测试程序，最后给出附加任务。此外，每个习题都配套教学视频。

本书是写给至少学过一门编程语言的读者的，本书有趣、简单，并且讲解方法独特，让读者了解众多 C 语言的基础知识和 C 程序中常见的缺陷，在慢慢增强自己的技术能力的同时，深入了解怎样破坏程序，以及怎样让代码更安全。

◆ 著 [美] 泽德 A. 肖（Zed A. Shaw）
译 王巍巍
责任编辑 杨海玲
责任印制 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
三河市君旺印务有限公司印刷

◆ 开本：800×1000 1/16
印张：20.75
字数：470 千字 2018 年 4 月第 1 版
印数：1-3 000 册 2018 年 4 月河北第 1 次印刷

著作权合同登记号 图字：01-2015-8786 号

定价：69.00 元

读者服务热线：(010)81055410 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

版权声明

译者简介

王巍巍是一名受软件和编程的吸引，从硬件测试做到软件测试，又从软件测试做到软件开发的IT从业人员。代码和翻译是他的两大爱好，此外他还喜欢在网上撰写和翻译一些不着边际的话题和文章。如果读者对书中的内容有疑问，或者发现了书中的错误，再或者只是想随便聊聊，请通过电子邮件（wangweiwei@outlook.com）与他联系。

译者序

大部分技术书籍都在教你一些具体的东西：某门语言、某种技术、某个框架、某个工具……你看完书，大致模拟演练一下，然后就可以把这样东西写在简历里。你把简历发给心仪的公司，就幸运地获得了一份新工作。你在新公司解决了一些技术问题，但也积累了一些技术债务。你解决的问题为你带来更多经验并铺平了你前进的道路，积累的债务你也不用担心，因为没过几年，你要么已经成功升职，没人敢让你还债，要么已经成功跳槽，没人能找到你还债了。

这就是每一名上进的程序员的上升之路，用一个时髦的词汇讲，这算得上个人成长的“增长黑客”之路。当然，等这本书出版的时候，这个词也许跟“给力”一样，被扫到互联网的垃圾堆里了。

这本书不是这样的，作者竟然找了一门已经过时的编程语言，来教你一些没法写在简历里的东西。作者疯了？

其实这本书和作者的其他书一样，表面上是在教你编程语言，实际上是在教你编程的思维方式和最佳实践。这些东西在学校的课堂里讲得不多，市面上的书籍讲这个的就更少。工作时间长了，你也许会遇到一种人，他们技术水平似乎挺不错，很多东西都能讲出些门道，但写的代码质量却不太理想。这样的人，也许就是缺了这么一课。这一课很多人都是在工作实战中通过栽跟头补上的，但是现在你可以通过这本书补上。

这本书的价值就是在于让初级水平的程序员也能写出牢靠可用的代码。沉下心打好基础，未来的路才会更顺畅。

关于这本书涉及的具体话题，请参考作者的前言“这不完全是一本 C 语言的书”，我就不在此赘述了。

在这里我要感谢人民邮电出版社编辑的辛勤劳动，并且感谢王以恒小朋友没有跟他爸爸我抢电脑玩。

如有问题或者建议，欢迎和我联系，我的邮箱是 wangweiwei@outlook.com。

致谢

我要感谢3类人，是他们让我的书以现在的面目问世：恨我的人、帮我的人，还有绘画的人。

恨我的人让我的这本书变得更强，更能站住脚。他们脑子不会转弯，盲目地崇拜着C语言的旧神祇们，完全没有教学经验。要是没有他们作为反例，我也许就不会这么努力，让这本教人成为更好的程序员的书完整问世了。

帮我的人包括Debra Williams Cauley、Vicki Rowland、Elizabeth Ryan，还有整个Addison-Wesley团队，以及每一个在线指出错误和提出建议的人。他们的制作、纠错、编辑、优化工作，让这本书变得更为专业，也更为优秀。

绘画的人就是Brian、Arthur、Vesta和Sarah，他们帮我找到了一种新的自我表达方式，让我忘掉了Debra和Vicki为我设得清清楚楚而我又次次赶不上的最后期限。没有绘画，没有这些艺术家们给我的艺术礼物，我的生活就没这么丰富和有意义了。

谢谢你们所有人在我写书过程中提供的帮助。完美的书不存在，这本书也谈不上完美，不过我已经尽力让它尽可能好了。

这不完全是一本 C 语言的书

别觉得自己上当了，这本书其实并不是一本教你 C 语言编程的书。你会学到编写 C 语言程序，但这本书给你最重要的一课是严谨的防御性编程。现在有太多的程序员会天真地假设他们写的东西没问题，但却总有一天，灾难性的失败会发生在这些东西上面。如果你主要学习和使用的都是现代编程语言来解决问题，那你尤其会碰到这种情况。阅读了这本书，并跟着做了里边的习题，你就会学会怎样写出有自卫能力的程序，它们可以防卫自己不被恶意行为和自身缺陷所伤害。

我使用 C 语言有一个很特别的原因：C 是一门“烂”语言。C 语言有很多设计选择在 20 世纪 70 年代还算颇有意义，但放到今天则毫无道理。从毫无限制到处乱用的指针，到严重没法用的 NUL 结尾的字符串，这些东西是 C 程序几乎所有安全缺陷的罪魁祸首。尽管 C 语言用途广泛，但我相信 C 语言之烂，它是最难写出安全代码的一门语言。我猜就连汇编语言都比 C 语言更容易写出安全代码。说句心底的大实话，我觉得大家都不该再写新的 C 代码了。

既然这样，那为什么我还要教你 C 语言呢？因为我想要让你成为一个更好、更强大的程序员。如果你要变得更好，C 语言是一个极佳的选择，其原因有二。首先，C 语言缺乏任何现代的安全功能，这意味着你必须更为警惕，时刻了解真正发生的事情。如果你能写出安全、健壮的 C 代码，那你就能用任何编程语言写出安全、健壮的代码。你在这里学到的技术，可以应用到今后你用到的任何编程语言中。其次，学习 C 语言让你能直接接触到如山似海的旧代码，还能教会你众多衍生语言的基本语法。一旦学了 C 语言，你学习 C++、Java、Objective-C 和 JavaScript 也就更容易，就连一些别的语言也会变得更加易学了。

告诉你这些不是为了把你吓跑，我计划把这本书写得非常有趣、简单而且“离经叛道”。我会给你一些用别的语言也许很少会去做的项目，通过这种方式让你从中获得乐趣。我会用我百试不爽的习题模式，让你学习 C 语言编程并且慢慢增强自己的能力，这种方式会让你觉得这本书很好上手。我还会教你怎样破坏程序以及怎样让你的代码变得更安全，让你知道为什么这些事情很重要，这种方式可以说是相当的“离经叛道”。你将学会怎样导致栈溢出和非法内存访问，了解众多 C 程序中常见的缺陷，最终知道自己面对的究竟是什么。

像我的所有书一样，这本书的通关很有挑战性，不过如果你真的通关了，那会成为一名更好且更自信的程序员。

未定义行为

当你学完这本书以后，你会有能力调试、阅读、修正几乎所有你遇到的 C 程序，还能在需

要的时候写出新的、稳固的 C 代码。然而，我不会教你官方的 C 语言。你会学到 C 语言，也能学到如何正确使用它，但官方 C 语言并不是非常安全。大多数的 C 程序员写的代码并不稳固，其原因就在于一个叫未定义行为（undefined behavior，UB）的东西。未定义行为是美国国家标准组织（ANSI）的 C 语言标准中的一部分，这部分罗列了编译器能忽略你写的代码的所有方法。这份标准里真的有这么一块内容，里边写着如果你这么写代码，那么编译器就不和你玩了，它的行为就会变得不可预测。当 C 程序读到字符串的结尾，就会发生未定义行为，这是 C 语言中极其常见的一种错误。再来讲点背景吧，C 语言将字符串定义为以 NUL 字节（为简化定义，可称为 0 字节）结尾的内存块。由于很多字符串都是来自程序之外，C 程序会经常接收到不包含 NUL 字节的字符串。当发生这种情况的时候，C 程序会越过字符串结尾接着读下去，读到计算机的内存中去，从而导致程序崩溃。C 语言之后的每一种语言都试图避免这种情况的发生，但 C 却是个例外。C 语言自己几乎完全不会预防未定义行为，而似乎每一个 C 程序员都认为这意味着他们无须应付这件事情。他们写的代码里充满了 NUL 字节越界的潜在可能性，当你指出这些地方以后，他们会说："不就是未定义行为嘛，没必要费劲儿预防的。"这种对 C 程序中大量未定义行为的依赖，就是大部分 C 代码都极其不安全的原因所在。

我写代码时会试图避免未定义行为，避免的方法就是让我写的代码要么不触发未定义行为，要么能防止未定义行为。后来我发现这是一个不可能的任务，因为未定义行为实在太多了，到处都是各种互相关联的陷阱，形成一个难解的戈耳狄俄斯之结。在你学习这本书的过程中，我会指出各种你会触发未定义行为的方法，告诉你可能的话该如何避免，以及如何在别人的代码中触发可能的未定义行为。不过你应该记住，要完全避免未定义行为这种带着近乎随机性质的东西是几乎不可能的，你也只能尽力而为。

> **警告** 你会发现 C 语言的铁粉会拿未定义行为这个话题来欺负你。有一类 C 程序员，他们写的 C 程序不多，但他们记住了所有的未定义行为，并以此来欺负编程新手的智商。别理这样的人。大部分时候，他们并不是真正会写程序的 C 程序员，他们目空一切，出言不逊，只会没完没了考你。他们所做的一切，都只是为了证明自己高人一等，而不是为了帮忙解决问题。如果你需要有人帮忙指点一下你的 C 代码，给我的邮箱 help@learncodethehardway.org 写封邮件就好了，我很乐意帮你。

C 是一门既美丽又丑陋的语言

未定义行为的存在，是你学习 C 语言的又一个理由，如果你想成为一名更好的程序员，这会对你很有帮助。如果你能用我教你的方法写出良好稳固的 C 代码，那你就能应付任何语言。C 语言还有积极的一面，它在很多方面都是一门真正优美的语言。尽管它功能强大，语法却简单到令人难以置信。在差不多 45 年里，众多语言都复制了 C 语言的语法，这不是没有原因的。

C 语言会给你提供很多，使用到的技术却极少。学完 C 语言后你会发现，这门语言既优雅美丽，同时又有几分丑陋。C 语言很老了，它就像一块纪念碑，远看雄伟壮丽，近看会有很多的裂缝和缺陷。

我知道我用的 C 挺稳固，因为我花了 20 年撰写整洁、稳固的 C 代码，它们支撑了大型程序的运作，而且基本没怎么出过问题。我的 C 代码支撑了 Twitter 和 Airbnb 等公司的业务，这些代码也许已经处理过数万亿个事务。它们极少出问题或受到安全攻击。在许多年里我的代码都支撑着 Ruby on Rails 的 Web 世界，它一直运行完美，甚至还防止过安全攻击，而别的 Web 服务器程序常被各种简单的攻击攻陷。

我的 C 代码编写风格是很稳固的，不过更重要的是我写 C 代码时的意识状态，这应该是每一个程序员都应具备的东西。我在着手 C 语言或者任何别的编程语言时，会时刻想着尽可能预防错误的发生，并且会带着凡事皆不会顺利的想法。别的程序员，就连那些据说很厉害的程序员，也会在写代码时假设一切顺利，而后则需要依赖未定义行为或操作系统来拉自己一把，这二者作为解决方案都是不及格的。你只需要记住，如果有人告诉你，说我在这本书里教你的不是“真正的 C 语言”，那么你可以考察一下他们的编程历史，如果他们的纪录没我这么好，那么没准儿你还可以用我教你的方法，给他们展示一下为什么他们的代码安全性不太好。

那么是不是我的代码就完美了呢？当然不是。这可是 C 代码。写出完美的 C 代码是不可能的，其实用任何语言写出完美的代码都是不可能的。编程的乐趣和烦恼，有一半都和这一点有关。我可以把别人的代码批得一文不值，别人也可以把我的代码贬得一无是处。所有的代码都是有缺点的，但我假设自己的代码总有缺陷，然后去避免这些缺陷，那么事情就不一样了。如果你学完了这本书，我给你的礼物就是教会你防御性编程的思维模式，这种意识在 20 年里为我带来了不少好处，让我写出了高质量、健壮的软件。

你会学到的东西

这本书的目的是让你掌握足够的 C 语言技能，从而可以自己写软件，或者修改别人的 C 代码。学完这本书以后，你应该去阅读 Brian Kernighan 和 Dennis Ritchie 的《C 语言编程设计（第 2 版）》，英文书名为 *C Programming Language, Second Edition*，这是 C 语言发明者写的一本书，又称作 K&R C。我将教会你的是以下内容：

- 基本的 C 语法和习惯写法；
- 编译、Makefile 和链接器；
- 找出 bug，防止它们发生；
- 防御性编程实践；
- 破坏 C 代码；
- 撰写基本的 Unix 系统软件。

等你完成了最后一个习题，你将会拥有充足的“弹药”，用来应对基本的系统软件、库以

及别的小型项目。

怎样阅读本书

本书针对的是至少学过一门编程语言的人。如果你还没学过编程语言，我建议你通过我的《“笨办法”学 Python》（*Learn Python the Hard Way*）开始学习，它是一本专为初学者写的书，是一本非常有效的编程入门书。读过《“笨办法”学 Python》以后，你就可以开始看这本书了。

对于已经学过编程的人来说，本书一开始也许看上去有些奇怪。别的书里边会有大段大段的讲解，然后让你时不时写一点儿代码。这本书不一样，每一个习题都有视频讲解，你一上手就要输入代码，然后我再向你解释你输入的内容。这种形式更为有效，因为用抽象的形式解释你不熟悉的东西你会很难理解，而如果你已经做过了一次，我解释起来就更为容易了。

由于本书结构独特，你必须在学习时遵守几条规则。

- 先看视频，除非习题中另有指示。
- 录入所有代码。禁止复制粘贴！
- 一字不差地录入代码，连注释也要一模一样。
- 运行代码，确保输出相同。
- 如果有缺陷，就修正它们。
- 做附加练习，不过如果遇到弄不清楚的东西，跳过去也没关系。
- 遇到问题先自己想办法解决，然后再求助。

如果你遵守这些规则，照着书里的内容去做了，最后还没有学会编写 C 代码，那么至少你已经尝试过了。C 语言并不适合每一个人，但尝试的过程会让你成为一个更好的程序员。

视频

本书为每一个习题都配了视频[①]，很多情况下一个习题会包含多个视频。这些视频至关重要，能让你完全领会本书的教学方法。这样做是因为撰写 C 代码的很多问题都是交互问题，交互的对象是失败、调试、命令等东西。Python 和 Ruby 这样的编程语言中，代码要运行就运行了。C 语言中要让代码运行，要修正问题，需要的交互要多得多。有些话题用视频讲解也更容易，比如指针和内存管理，在视频中我可以演示机器真正是如何工作的。

我建议你先看视频再做习题，除非我另有指示。在有的习题中，我使用一个视频演示问题，再用另一个视频来演示解决方案。在另外的大部分习题中，我使用视频作为讲座，然后你完成练习，弄懂课题。

① 本书配套视频读者可扫各习题首页标题边上的二维码在线观看。——编者注

关键技能

我猜你是从一门菜鸟语言来到这里的（我只是在调侃而已，你应该看得出来）。要么你来自像 Python 或者 Ruby 这样“还算能用”的语言，这些语言让思维不清、半吊子、瞎鼓捣的人也能写出能运行的程序来；要么你用过 Lisp 这样的编程语言，这些编程语言假装计算机是某个纯函数的仙境，四周还装了五彩的婴儿墙；要么你学过 Prolog，因而认为整个世界应该只是一个供你上下求索的数据库。更糟糕的还在后面，我打赌你还用过某个集成开发环境（integrated development environment，IDE），所以你的脑子充满里记忆空洞，如果你不是每敲 3 个字符就按一次 Ctrl+Space 的话，我怕你连一个完整的函数名称都敲不出来。

不管背景如何，你都可能有以下 4 样技能有待提高。

读写能力

如果你平时使用 IDE 的话，尤其如此。不过大体来说我发现程序员略读的时候太多了，从而导致理解性阅读能力有些问题。他们将代码扫视一遍就觉得自己读懂了，其实不然。其他编程语言还提供了各种工具，从而避免让程序员直接撰写代码，所以一旦面对 C 这样的编程语言，他们就立马崩溃了。最简单的办法就是要理解每个人都有这样的问题，解决方案就是强迫自己慢下来，更加细致地去读写代码。一开始你也许会觉着很痛苦、很烦躁，那就增加自己休息的频率，最后你会觉得这其实也很容易做到。

关注细节

这方面没有人能做得好，这也是产生劣质软件的最大成因。其他编程语言会让不够专注的你蒙混过关，但 C 语言却要求你完全聚精会神，因为 C 语言直接和计算机打交道，而计算机又是极其挑剔的。在 C 的语境中没有“有点儿像”或是“差不多”这样的说法，所以你需要专注。反复检查你的工作。在证明正确之前，要先假设一切都可能是错的。

发现差异

用过其他编程语言的程序员有一个问题，就是他们的大脑已经被训练成可以发现那种语言中的差异，而不是 C 语言中的差异。当你在对比你的代码和标准答案时，你的视线会直接跳过那些你认为不重要或不熟悉的部分。我给你的解决办法是：强迫自己观察自己的错误，如果你的代码跟本书中的代码不是一字不差，那它就是错的。

规划和除错

我喜欢其他更简单的编程语言，因为我可以“胡搞乱来”。我把想法敲出来，然后就能直接在编译器里看到结果。这些语言可以让你很方便地尝试新的想法，但你有没有发现：如果你一直用“乱改直到能用”的方法写代码，到头来就是什么都不能用了。C 语言对你要求比较高，因为它要求你先计划好要创建的东西。当然你也可以偶尔胡乱弄弄，但和其他编程语言相比，你需要在更早的阶段就开始认真做计划。在你写代码之前，我会教你如何规划程序的关键部分，希望这能同时使你成为一个更优秀的程序员。即使是很小的计划也能让你的后续工作更为顺利。

在学习 C 语言的过程中，你将被迫更早、更多地应对这些问题，所以学习 C 语言更能让你成为一名更好的程序员。你不能对自己写的东西思维不清，否则什么都不会做出来。C 语言的优势是，作为一门简单的语言，你可以自己把它弄明白，因此如果你要学习机器的工作原理，并增强这些核心的编程技能的话，C 语言是上佳的选择。

目录