Broadview
WWW.BROADVIEW.COM.CN

WRITE GREAT CODE
VOLUME 2: THINKING LOW-LEVEL, WRITING HIGH-LEVEL

编程卓越之道

第二卷：

运用底层语言思想编写高级语言代码

[美] Randall Hyde 著
张菲 译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

NO STARCH PRESS

编程卓越之道

第二卷：运用底层语言思想编写高级语言代码

WRITE GREAT CODE

VOLUME 2: THINKING LOW-LEVEL, WRITING HIGH-LEVEL

[美] Randall Hyde 著

张 菲 译

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书是《编程卓越之道》系列书的第二卷，将探讨怎样用高级语言（而非汇编语言）编程得到高效率机器代码。在书中，您可以学到如何分析编译器的输出，以便检验代码的所作所为，从而得到高质量的机器码；了解编译器为常见控制结构生成的典型机器指令，以便在编写高级语言程序时选用恰当的语句；掌握编译器将各种常量和变量类型转换成机器数据的方法，以便于使用这些数据写出又快又短的程序。

书中的理论超出了特定的编程语言和CPU架构，以各种处理器平台进行开发的高级语言程序员都能从中汲取到卓越编程的营养。

版权贸易合同登记号　图字：01-2005-4463

图书在版编目（CIP）数据

编程卓越之道. 第二卷，运用底层语言思想编写高级语言代码 /（美）海德（Hyde,R.）著；张菲译.
—北京：电子工业出版社，2007.4
书名原文:Write Great Code Volume 2：Thinking Low-Level，Writing High-Level
ISBN 978-7-121-04125-9

Ⅰ. 编… Ⅱ.①海…②张… Ⅲ.程序设计 Ⅳ.TP311.1

中国版本图书馆 CIP 数据核字（2007）第 040383 号

责任编辑：周　筠　杨绣国
印　　刷：北京智力达印刷有限公司
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：40.75　字数：810 千字
印　　次：2007 年 4 月第 1 次印刷
定　　价：69.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
服务热线：（010）88258888。

人们对《编程卓越之道 第一卷：深入理解计算机》的赞誉之辞

倘若你没有受过正规培训，或者缺乏高手指点，你会对 Randall Hyde 的《编程卓越之道》系列书感兴趣……这一卷的前 5 章及布尔逻辑的那章使该书物有所值。

——UNIX REVIEW

该书详细讲解了大多数程序员认可的东西。

——[英] COMPUTER SHOPPER

该书读起来饶有味道。

——VSJ MAGAZINE

本书在一个相当复杂的层次上详述了机器的内部工作原理，它不是为新手准备的。程序员若对此层次上的编程有兴趣，将会发现该书堪称良师益友。

——SECURITYWORLD.COM

它填补了计算机科学学位读物的空白，也完全应该成为攻读该学位的必修书籍……读了此书后，你就会对所写的高效代码有更透彻的理解和把握，也会清楚自己该如何写出高效代码。

——MACCOMPANION，在五星级评价中给本书打了 5 分

对于那些渴望写出高品质代码，又不想学习汇编语言的人来说，《编程卓越之道 第一卷：深入理解计算机》可谓必读书籍。

——WEBSERVERTALK

Hyde 谈的乃是所有软件开发核心、计算机架构基石的话题。

——PRACTICAL APPLICATIONS

这难道不是你自学编程的最佳教材吗……它对各种语言，以及编程经验的所有层次都有用……赶紧去买一本好好读读吧！

——BAYLISA（港湾大型安装系统管理员协会）

译　序

比起老前辈来说，我们这一代程序员要幸福得多——我们可以从几百种高级编程语言中挑选自己顺手的那种来用，可以坐在显示器前操作键盘、摆弄鼠标，使用各式先进的输入输出设备。机器的处理和联网速度如此之快、硬盘如此之海量、文本编辑器的功能如此之强大、集成开发环境又是如此方便，以至于我们只需用鼠标指指点点，偶尔敲几下键盘，就能得到一个像模像样的应用程序。大行其道的高级语言一方面屏蔽了底层硬件工作的细节，降低了编程的难度，大大加快了程序的生成速度；另一方面也使得大多数新一代的程序员不了解代码的底层实现，很容易忽略代码的效率问题。而汇编语言程序员由于以机器指令形式与硬件打交道，所以他们对指令的执行效率了解得都很透彻。可是众所周知，汇编语言是一种底层语言，效率虽高但可读性差，编写、维护都很不容易，且没有可移植性。怎样才能既利用高级语言开发周期短、维护便捷的优势，又不过多地丧失效率，逼近汇编语言程序的性能呢？这是每个专业程序员都应认真思考的问题。

如今计算机的资源是如此地丰富，让许多程序员觉得程序效率低一些也没什么大不了的。这种想法对于编写简单程序的非专业程序员尚能说得过去，但实际上，商品级软件的功能在越来越强的同时也在变得越来越复杂，必须耗用更多的计算机资源才能运行。我们经常感到恼火，计算机（硬件）性能已经够强了，为什么（软件）运行却更慢了呢？那就是因为许多时候软件运行了一大堆不必要的进程或功能，或者算法不够精良，耗用了太多的资源所致。

《编程卓越之道》系列书旨在讲解卓越编程的方法。具体到第二卷，则是探讨怎样用高级语言（而非汇编语言）编程，得到高效率机器代码。它将引导我们在确保实现设计目标的前提下，学习如何尽量少占用系统资源，又充分发挥代码的性能。该卷将使我们鱼与熊掌兼得，既让我们免于用汇编语言编程之苦，免于牺牲程序的可读性、可维护性和可移植性，又能通过高级语言得到尽量高效的机器码。其核心思想就是“Thinking

Low-Level, Writing High-Level”，即用高级语言编程的同时，思考其底层机器码会是怎样的。它不是一本入门级的编程教材，而是我们用高级语言编程获得了一些经验后，希望能再提高自己水平的用书。尽管其中的例子多数都采用 C/C++、Pascal 语言，汇编用的是 HLA 等汇编语言，但书中的理论超出了特定的编程语言和 CPU 架构，以各种处理器平台开发的高级语言程序员都能从中汲取到卓越编程的营养。

本书为中文版《编程卓越之道》第二卷，基于原著此卷第一次印刷的版本（2003017502）翻译而成。翻译尽量承袭第一卷的风格，但力图更上一层楼：为了方便读者查阅书中提到的参考资料，我特意搜索了这些原版书籍在国内的引进版，以“译者注”的形式标出了其中文版或影印版的书名、译者、出版社和书号。有引进版的书仅在第一次出现时标注原英文书名并辅以译者注，此后所有出现此书的地方都标识最新版的中文书名；没有查到引进版的书则始终以英文书名标识。读者可据此了解该书的引进情况。本卷也将索引做了翻译，并按拼音排序，以便用户查找书中内容。另外，若干术语的译法与第一卷有所区别，主要包括：padding byte（s）在第一卷中译作“补齐字节”，本卷则译为“填充字节”；architecture 在第一卷中译作“体系结构”，本卷则主要译为“架构”；object 在第一卷中大都译作“对象”，尽管该单词确为此意，但考虑到这样容易让读者与“面向对象编程”中的对象混淆，因此除了第 12 章在讲述类时把 object 译作“对象”或目标（码、文件...）之外，其余位置均根据上下文语境改译为“数据”、“变量”等。

原书中出现单词“teach”、“you”的地方比比皆是，在翻译过程中我觉得这种语气对读者不够妥当，因此在译文中尽量以“我们”代替“you”，而将“teach”改译为“研究、探讨、研讨、讨论、学习”一类的词汇。另外原文的叙述用了大量的括号，严重影响了阅读的流畅性，因此在翻译过程中我设法除去了大部分括号，将括号内的信息融入正文。所有对原著的改动都已经过作者确认或听从作者建议。比较重要的修改则以“译者注”标于所在页面；至于单词拼错等细微错误，则直接纠正到正文中。还有，原著第 3、9、12 章的目录结构不尽合理，所以本书应作者要求调整了这些章的目录，对此不再另加“译者注”。

翻译本书的过程也是我的学习过程，它使我对编译器的优化原理以及编程时的注意事项有了更加理论化的认识，相信对渴望提高编程水平的程序员来说本书会是一部很好的学习资料，必然能从中收获甚丰。

在翻译本书的过程中我遇到了一些困惑，曾试图直接向 Randall Hyde 先生和 No Starch 出版社求教，但很久都没有回音，后在电子工业出版社刘皎老师的帮助下才与之取得了联系，消除了翻译遗留的问题。在此向刘皎老师致敬，并向原著作者和出版社致以谢意。陈元玉编辑作为本书的前期联络人，为我提供了周到细致的服务；译稿成文后由杨绣国

编辑作了全方位的审校，指出了不少问题，对提高本书的翻译质量起了很大的作用；周筠老师把这本书的翻译机会交给了我。她的人格魅力给我留下了深刻的印象，是我学习做人的榜样。在此我向他们表达由衷的感谢、感激之情。我的家庭，母亲王华敏、妻子雷顺和刚出生的儿子张乐康是我事业成长的动力，正是你们的照顾和鼓励，使我有条件完成此书的翻译工作。

最后也是最重要的，我想感谢选择阅读本书的读者。市面上并非仅此一部讲述编程优化的著作，而且每个人的时间和精力都很宝贵，您愿意研读本书，愿意为它投入时间和精力，表明了您对它的信任和期望。我希望本书能帮助您达到目标，祝您成功！

译文力争以通俗顺畅的汉语再现原著的知识。由于译者水平有限，可能存在某些疏漏之处，请读者不吝赐教。您的意见、建议能够帮助我们改善本书的质量。也欢迎你发邮件到 zhangfei97@163.com，与我交流本书相关的信息，再次感谢！

张菲

2007 年 3 月

致　谢

这本书的内容原本是想当作《编程卓越之道 第一卷：深入理解计算机》的最后一章。第一卷的策划编辑 Hillel Heinstein 觉得作为一章太长，何况内容与第一卷的书名也不太切题。于是我们决定将这些材料扩充成为单独的一卷，所以 Hillel 是我为本书问世要感谢的第一个人。

当然，将长达 200 页的一章变成完整的书籍还要做大量的工作，很多人参与了本书的创作过程。我想专门感谢这些为本书做出贡献的人。

感谢我的好友 Mary Philips，是您帮我整理了《汇编语言编程艺术》“The Art of Assembly Language”和本书的书稿，使一些材料各得其所；

感谢出版人 Bill Pollock，您深信本系列书的价值所在，给予了我指导和精神支持；

感谢产品经理 Elizabeth Campbell，您是我在 No Starch 的主要联系人，引领着这个项目的进展，直至它成为现实；

感谢编辑 Kathy Grider-Carlyle，是您确保了本书文法上的正确性；

感谢策划编辑 Jim Compton，您花费了相当多的时间来改进本书的可读性；

感谢 Stephan Provines，是您在校对过程中发现了一些排版和印刷方面的错误；

感谢 Riley Hoffman，您负责处理页面排版等繁杂事务，帮助确保本书内容（尤其是代码）能让人看懂；

感谢 Christina Samuell，您也负责本书的排版，还为我在创作上帮了不少忙；

感谢技术审阅 Benjamin David Lunt，是您保证了本书的技术质量；

感谢 Leigh Poehler 和 Patricia Witkin，两位负责着这本书的销售和市场策划工作。

我还要向 No Starch 出版社以前的编辑 Susan Berge 与 Rachel Gunn 致谢。尽管在本书最终出版时您到了别的地方，但对本项目的功劳是不可抹杀的。

最后，我想把这本书送给我的侄子侄女们：Gary、Courtney（Kiki）、Cassidy、Vincent、Sarah Dawn、David 和 Nicholas。我猜你们一定会很开心，因为书里印有自己的名字。

引　言

卓越代码涵盖很多方面的要素，远不是一本书能够包容的。所以本书作为《编程卓越之道》系列书的第二卷，将关注卓越代码的一个重要组成部分——性能。随着计算机系统的速度从原先的兆赫级增长到几百兆赫，又到上吉赫兹，软件获得了广阔的性能施展空间。时至今日，软件工程师声称“代码根本无需优化”已是司空见惯的现象。有趣的是，使用计算机应用程序的人却很少说这样的话。

尽管本书讲述的是如何编写高效代码，但它并非一本关于优化的书。优化是在软件开发周期几近结束时，由软件工程师判断其代码为何不能符合性能要求，并为达到要求而修改代码的过程。然而不幸的是，倘若直到优化阶段才想到应用程序的性能，优化将无从实施。在软件开发周期的设计与实现阶段，就该确保应用程序具备合理的性能特征。优化措施可以调校系统性能，但对编写糟糕的代码也是无力回天。

Tony Hoare 最早说过："不成熟的优化乃万恶之源"，而 Donald Knuth 让这句话流行开来。它成了长期以来有些软件工程师的战斗口号，这些工程师不到软件开发周期行将收尾，就总是一味逃避考虑应用程序的性能，即便最后也往往以经济或销售时限为借口而不了了之。然而，Hoare 可没说"在应用开发早期阶段，关注应用程序的性能就是万恶之源"。他强调的是"不成熟的优化"，在当时这意味着注意汇编语言代码的周期数和指令条数，而不是在初始程序设计期间要操心编码的类型，毕竟此时程序骨架尚未定型，故而 Hoare 的话是切中要害的。下面这段话摘自 Charles Cook 在 *http://www.cookcomputing.com/blog/archives/000084.html* 的一篇短文，其中谈到了某些人对上面说法断章取义的问题。

> 我经常在想，这句话老是导致软件设计者犯严重错误，因为不同立场的人对其有不同的解读。
>
> 这句引语的完整版本是"我们应当在 97%的时间里忘掉琐碎的效率问题：不成熟的优化乃万恶之源。"我赞同这种说法。在性能没有明显成为瓶颈时，花大量时间去精雕细刻代码是不值得的。但相反，要是在系统级进行软件设计，着手时就应考虑性能问题。出色的软件开发者会自动这样做，他们已经形成了"性能问题终将导致麻烦"的意识；而没有经验的开发人员却不这么想，他们自以为后期调整能够将以前遗留的问题统统排除。

Hoare 真正的意思是，软件工程师应当把心操在诸如精良的算法设计、算法的恰当实现等方面，然后再注重传统的优化措施，例如执行特定语句要花费多少 CPU 周期之类的问题。

我们当然可以将本书的概念用到优化阶段，但多数技术其实都需要在初始编码时运用。如果在程序行将完工时才付诸实施，那么很可能在软件中已无其立足之地，因为在既成事实面前，履行这些思路要做的工作量太大了。

本书意在探究怎样选择适当的高级语言语句，以便让现代的优化型编译器生成有效率的机器码。我们在用高级语言编程时，达到某种结果可以采取许多种途径，而在机器级有些方法天生就比另一些强。尽管舍弃高效的语句，选用欠效率的语句序列可能有充分理由——例如出于可读性考虑——但实际情况是，多数软件工程师并不了解高级语言语句的运行开销。没有这些知识，他们在选择语句时就无从作出明智的决定。本书的目标就在于改变这种面貌。

有着丰富经验的软件工程师大概会争辩说，这些技术都只会对性能起到微不足道的改善作用。有些情况下这种评价是对的。但我们必须知道，这些微小的改良措施具有累加效应。人们当然能乱用本书建议的技术，生成难以看懂和维护的代码。但本书的真正意义在于，从系统设计角度看，当我们有两种等效的代码序列可选时，就该选择其中更有效率的那个。遗憾的是，当今的许多软件工程师都不知道哪种实现方案能够产生更高效的机器码。

通过研究编译器输出——在本书里就是这么做的——我们不必成为汇编语言专家就能写出高效的代码，但我们至少得懂一些汇编语言的知识。第 3 章和第 4 章分别给出了 80x86、PowerPC 汇编语言的入门教程。

在第 5 章和第 6 章中，我们将学习通过检查编译器输出，来确定所用高级语言语句的代码质量。这两章描述反汇编器、目标代码转储工具、调试器、高级语言编译器显示汇编代码的各种选项，以及一些有用的软件工具。

本书的其余部分，从第 7 章到第 16 章，说明了编译器对不同高级语言语句、数据类型生成机器码的原理。有了这些知识来武装头脑，我们就能选取最适当的数据类型、常量、变量和控制结构，以便生成高效率的应用程序。

当我们看这本书的时候，要牢记 Hoare 博士的话——“不成熟的优化乃万恶之源”。倘若你不在乎本书里的信息，生成了难以看懂和维护的代码，当然可能也会工作。但在项目设计和实现的早期，代码还处于漂忽不定、可塑性很大的阶段，这么做是极其危险的。不过要知道，本书并非不分青红皂白地选取最高效的语句序列，它讲解了若干高级语言程序构建方案的开销，在我们有选择余地时，从而可以做出该用哪种序列的明智决定。有些时候出于恰当的理由，会选择欠效率的语句序列。然而，倘若我们连给定语句的开销都不清楚，又怎能选择较高效的方案呢。

有兴趣的读者还可以阅读其他有关“万恶之源”的文章，请查看下列网页：

http://blogs.msdn.com/ricom/archive/2003/12/12/43245.aspx

http://en.widipedia.org/wiki/Software_optimization

很抱歉这些网址可能并不一直有效。

目录一览

目　录