<<代码之美>>

图书基本信息

书名：<<代码之美>>

13位ISBN编号：9787111251330

10位ISBN编号：7111251334

出版时间：2008年09月

出版时间：机械工业出版社

作者：Grey Wilson

页数：599

译者：聂雪军

前言

序Greg Wilson我在1982年夏天获得了第一份程序员工作。
在我工作了两个星期后，一位系统管理员借给了我两本书：Kernighan和Plauger编写的《The Elements of Programming Style》(McGraw-Hill出版社)和Wirth编写的《Algorithms + Data Structures = Programs》(Prentice Hall出版社)。
这两本书让我大开眼界——我第一次发现程序并不仅仅只是一组计算机执行的指令。
它们可以像做工优良的橱柜一样精致，像悬索吊桥一样漂亮，或者像George Orwell的散文一样优美。
自从那个夏天以来，我经常听到人们感叹我们的教育并没有教会学生看到这一点。
建筑师们需要观摩建筑物，作曲家们需要研习他人的作品，而程序员——他们只有在需要修改bug时才会去阅读其他人的代码；即使在这个时候，他们也会尽可能减少阅读量。
我们曾告诉学生使用有意义的变量名，曾向他们介绍过一些基本的设计模式，但很奇怪，为什么他们编写的大多数代码都是很难看的呢！
本书将试图改变这种状况。
2006年5月，我邀请了一些著名的（以及不太著名的）软件设计师来分析和讨论他们所知道的漂亮代码。
正如在本书中将要介绍的，他们在许多不同的地方发现了代码的漂亮性。
有些漂亮性存在于手工精心打造软件的细微之处，而有些漂亮性是蕴涵在大局之中——那些使程序能够持续发展的架构，或者用来构造程序的技术。
无论他们是在什么地方发现的这些漂亮性，我都非常感谢我们的投稿人抽出时间为我们奉献了这样的一次学习旅程。
我希望你能够享受阅读此书的乐趣，就像Andy和我非常享受编辑这本书的过程，此外，我还希望这本书能激发你创建出一些漂亮的作品。
在学术界，有一种叫“论文集”的东西，把许多人的论文合到一起出版，让读者能够在一本书的篇幅之内，了解某个特定领域的研究状况，是有效的知识传播手段之一。
技术界比较少见到类似的出版物，的确是一种遗憾。
《代码之美（Beautiful Code）》就是这样一本书。
三十三位大牛人的技术文章汇集到一起，讲述作者们认为“最漂亮的代码”，涉及应用领域虽广，而代码之美却一以贯之。
如果我们 承认编程是门艺术——具有高度创造性和人类智慧参与的活动，不是艺术是什么？
——那么，这几十篇文章就体现了这门艺术中最高的境界。
别担心！
大牛们可不是坐而论道，也没有写什么常人不可索解的奥义，文章主题之朴实无华，比如“查找”，比如“分布式编程”，比如“Linux内核驱动模 型”……几乎要让人以为是不知道什么人编写的大学教材呢。
这貌似普通的三个主题，作者分别是XML创始人之一Tim Bray、Google Map/Reduce架构发明人Jeff Dean和Linux内核维护者Greg Kroah-Hartman——吓死人的阵容。
其余文章也都类似，小题目中见大手笔。
我深信这帮大牛接受约稿、写这种“小”文章，实在是出于对编程的热爱，出于对“漂亮代码”的不懈追求。
所谓“漂亮代码”，意思远超“规范、好看”，更多地体现出逻辑、思路与架构。
一万块 最漂亮砖头堆出来的，不一定是大厦。
建筑师在建造大厦之前，胸中早有蓝图在。
《代码之美》，正展示了三十三位最优秀建筑师胸中的蓝图。
这本书能出中文版，是中国程序员的福音，其中的每篇文章，都值得读者细细咀嚼、回味。
我已经迫不及待，想要看到正式印刷的版本了。
韩磊（CSDN副总经理）向大师学习美潘加宇UMLChina首席执行官我1989年参加高考时，总分120分

的语文才考了71分，这最弱一环差点造成致命打击。
最害怕的是写作文，记得当年的题目是写一封信，在右江盆地高温的教室里，我写得汗都滴在试卷上。
没想到，后来看的东西多了，“不会作诗也会吟”，居然也能接杂志和出版社的约稿，写一些文章和序言之类。
翻开本书第29章，Ruby之父Yukihiro Matsumoto（松本行弘）说：Treating Code As an Essay。
写代码如同写文章一般，多看多研究大师的作品，才能够信手拈来，写出美丽的代码。
本书就是33位大师的倾情之作。
本书并不限定于讨论某一种语言的技巧，你能看到大师们使用现在流行的Java、Ruby，也能看到历史悠久的Fortran。
讨论的领域从Linux内核，到NASA火星探测器、ERP系统，让我们从不同角度来体会代码之美。
美除了让人欣赏，还能带来金钱。
因北京奥运的成功，博尔特的速度之美估计价值上千万美元。
美丽的软件也一样。
一些软件公司，公司人少且稳定，多年来专注于做某一个小领域里的软件，对软件的打磨可谓是精雕细琢，美丽软件带来的利润自然也很可观。
比起那些靠低人力成本、低价格在市场上打拼的“程序员民工”公司，他们要活得滋润得多，安全得多。
享受代码之美“希望写出漂亮代码的开发者可以向艺术家们学习一些东西。
画家常常放下手中的画笔，然后远离画布一段距离，围着它转一转，翘起脑袋，斜着看看，再从不同的角度看看，在不同的光线下看看。
在寻求美的过程中，他们需要设计这样一些视角并使它们融为一体。
如果你的画布是个集成开发环境（IDE）而你的媒介就是代码，想一想，你如何做到离开画布一段距离，用挑剔的眼光从不同的视角来审视你的作品？
——这将使你成为一个更优秀的程序员，并帮你写出美丽的代码。
”写这段话的Alberto Savoia在他的文章里真的没有讲什么令人敬畏的高技术或是大架构，他讲的是每个计算机系的大二学生都熟悉的二分查找。
所以Savoia真的是在讲如何写出漂亮的代码，所以才选择了这么一个所有人都清楚得不能再清楚的例子。
你会觉得这种事情都是些不谙世事的小程序员才会热衷于干的吧？
可这位Savoia却是从Google离职以后开创了Agitar Software公司（http://www.agitar.com/）的不折不扣的创业者。
有意思吗？
一个胡须花白、在这个行业里厮混了数十年、拥有自己公司的老家伙，还在乐此不疲地谈论“漂亮的代码”。
这本《代码之美》就是由三十多篇像这样有意思的文章组成的。
像Brian Kernighan、Tim Bray、Charles Petzold、Douglas Schmidt、Yukihiro Matsumoto这样的名字，你甚至很难想象他们同时出现在同一本书上。
或许也只有“漂亮的代码”这样的话题才能激起他们共同的兴趣。
于是就有了这本了不起的书：从正则表达式匹配器到图像处理，从通信到基因排序，这些可能是世界上最优秀的程序员毫不吝啬地向读者展示：不论面对什么问题、使用什么语言，代码的美感都是始终存在的，而且这种美感应该是程序员毕其一生不懈追寻的。
作为《重构》的译者，不时有人会问我一些关于重构的问题，其中一个问题让我最感为难：为什么要这样做？
真的，如果不是要修改代码，也不是要添加功能，为什么要把这段代码抽取出来呢？
让每个方法都保持5行以内的长度到底有什么好处呢？
这种时候与其说是有什么利弊权衡，毋宁说就是为了让代码“更漂亮”。

当然了，在大部分时间里，软件开发是一项集合了科学、工程和服务的工作，但——至少在我们的内心深处——它多少还有那么一点艺术的成分。
除了完成任务以外让自己手上的代码更具美感，也算是对自己作为程序员的梦想的小小坚持吧。
所以，既然你已经拿起了这本书，就暂时放开那些功利的目标吧——别误会，这可不是一本没用的书，通过阅读这些“高手”们的编程心得，对自己的能力提升就算不能立竿见影至少也有潜移默化之功。
但那也只是装珍珠的盒子而已。
在一个安静的周末，给自己泡上一杯清茶，跟着三十三位顶尖高手畅游在代码世界，在他们的指引下遍赏代码之美，这才是作为一个程序员最大的享受呢。
熊节ThoughtWorks咨询师等度的流明——代码之美?序一我上一次印象深刻的美的体验，大概已经是在十年之前了，那只是在午后睡醒，面对窗外的一棵大梧桐树时的感觉。
不过这并不是说我这十年来都只看到了丑的事物，而是说我已经忘了去观察既已存在的美。
直到我拿到这本《代码之美》，我忽然地回到了那种仰望着星星烁耀的夜空，或低头沉思于一两句大家文字的日子里。
那时刻我既不是在思考，也不是在分析，更不是在解释，而只是在感受自然的、文字的，或将自然蕴于文字之中的，美。
二有一本书开启了一个时代，而我们如今仍然在这个时代之中而不知觉于这本书的深远影响，那是三位图灵奖得主合著的《结构程序设计》(1)。
其中Dijkstra将人“理解一个程序的种种思维方法”归为三种：枚举、数学归纳和抽象。
显然Wirth先生更为深层地看到了程序的本质，他说“程序＝算法+数据结构”(2)。
他揭示了这样一个事实：一个未知的、无序的世界是不可能实现“程序”的，于是我们抽象它——使它成为结构，或者对象，或者网，或者某个相对规则的事物。
然后，我们再着之以“算法”。
《代码之美》这本书，三十八位大师，在三十三章的内容中详细讨论了代码中抽象的过程、算法的过程和编程的过程。
显然的，这些正是程序中最深刻的美。
如同花之蕊，叶之脉，以及维系花蕊叶脉的美的，那些汁液。
这种对美的触及，使他在我面前闪耀着与前两本书等度的流明。
三“只有在不仅没有任何功能可以添加，而且也没有任何功能可以删除的情况下，设计师才能够认为自己的工作已臻完美。
”(3)然而编程的过程呢？
我们最初只是想实现一个功能。
但为了实现它，我们写了一段功能代码、一段测试代码、一段功能代码的配置代码，一段功能代码的配置代码的测试代码……如此往复不休。
我们回到原始的问题，原本只是要做一个“实现某项功能”的代码，我们却为何把代码做到了“往复不休”的绝地？
或者你做的事情并不完美，但是你应该知道所谓完美的终极。
代码要不停的测试，以及为测试代码再写测试代码，这一过程也不是美的。
或许你认为它“必须”，但你应知道它终究不美。
四大师们也并没有创生完美的能力，他们只是在一步步地进行着。
在这本书里，Adam Kolawa告诉你的，Lincoln Stein告诉你的，以及Elliontte Rusty Harold等等告诉你的，就是那经年累月地或亦步亦趋地进行过程，和那个“终极完美”的定义。
这只是过程和隐于过程中对美的追求。
而“美”是什么，还是在你的心底。
你心中原本就没有美的感受，如何写得出美的代码？
所以代码写到烂处，写到心胸滞涩处，便不如寻一清静所在，捧《代码之美》一册，回顾一下，品味一下，吐故纳新一下了。

看得多远，取决于你站得多高；要想成为他们，你得先知道他们。
这就是我的建议了。

<<代码之美>>

内容概要

《代码之美》介绍了人类在一个奋斗领域中的创造性和灵活性：计算机系统的开发领域。
在每章中的漂亮代码都是来自独特解决方案的发现，而这种发现是来源于作者超越既定边界的远见卓识，并且识别出被多数人忽视的需求以及找出令人叹为观止的问题解决方案。
《代码之美》33章，有38位作者，每位作者贡献一章。
每位作者都将自己心目中对于“美丽的代码”的认识浓缩在一章当中，张力十足。
38位大牛，每个人对代码之美都有自己独特的认识，现在一览无余的放在一起，对于热爱程序的每个人都不啻一场盛宴。
虽然《代码之美》的涉猎范围很广，但也只能代表一小部分在这个软件开发这个最令人兴奋领域所发生的事情。

作者简介

全球38位顶尖高手、众多语言之父经典之作

书籍目录

章节摘录

第3章 我从未编写过的最漂亮的代码Jon Bentley我曾经听一位大师级的程序员这样称赞到，“我通过删除代码来实现功能的提升。
”而法国著名作家兼飞行家Antoine de Saint-Exup é ry的说法则更具代表性，“只有在不仅没有任何功能可以添加，而且也没有任何功能可以删除的情况下，设计师才能够认为自己的工作已臻完美。
”某些时候，在软件中根本就不存在最漂亮的代码，最漂亮的函数，或者最漂亮的程序。
当然，我们很难对不存在的事物进行讨论。
本章将对经典Quicksort（快速排序）算法的运行时间进行全面的分析，并试图通过这个分析来说明上述观点。
在第一节中，我将首先根据我自己的观点来回顾一下Quicksort，并为后面的内容打下基础。
第二节的内容将是本章的重点部分。
我们将首先在程序中增加一个计数器，然后通过不断地修改，从而使程序的代码变得越来越短，但程序的功能却会变得越来越强，最终的结果是只需要几行代码就可以使算法的运行时间达到平均水平。
在第三节将对前面的技术进行小结，并对二分搜索树的运行开销进行简单的分析。
最后的两节将给出学完本章得到的一些启示，这将有助于你在今后写出更为优雅的程序。
3.1　我编写过的最漂亮代码当Greg Wilson最初告诉我本书的编写计划时，我曾自问编写过的最漂亮的代码是什么。
这个有趣的问题在我脑海里盘旋了大半天，然后我发现答案其实很简单：Quicksort算法。
但遗憾的是，根据不同的表达方式，这个问题有着三种不同的答案。
当我撰写关于分治（divide-and-conquer）算法的论文时，我发现C.A.R. Hoare的Quicksort算法（“Quicksort”，Computer Journal 5）无疑是各种Quicksort算法的鼻祖。
这是一种解决基本问题的漂亮算法，可以用优雅的代码实现。
我很喜欢这个算法，但我总是无法弄明白算法中最内层的循环。
我曾经花两天的时间来调试一个使用了这个循环的复杂程序，并且几年以来，当我需要完成类似的任务时，我会很小心地复制这段代码。
虽然这段代码能够解决我所遇到的问题，但我却并没有真正地理解它。
我后来从Nico Lomuto那里学到了一种优雅的划分（partitioning）模式，并且最终编写出了我能够理解，甚至能够证明的Quicksort算法。
William Strunk Jr.针对英语所提出的“良好的写作风格即为简练”这条经验同样适用于代码的编写，因此我遵循了他的建议，“省略不必要的字词”（来自《The Elements of Style》一书）。
我最终将大约40行左右的代码缩减为十几行的代码。
因此，如果要回答“你曾编写过的最漂亮代码是什么？
”这个问题，那么我的答案就是：在我编写的《Programming Pearls, Second Edition》（Addison-Wesley）一书中给出的Quichsort算法。
在示例3-1中给出了用C语言编写的Quicksort函数。
我们在接下来的章节中将进一步地研究和改善这个函数。
【示例】 3-1 Quicksort函数void quicksort（int l, int u）{int i, m;if （l >= u） return;swap（l, randint（l, u））;m = l;for （i = l+1; i

后记

后记Andy Oram《Beautiful Code》介绍了人类在一个奋斗领域：计算机系统的开发领域中的创造性和灵活性。
在每章中的漂亮代码都来自独特解决方案的发现，而这种发现来源于作者超越既定边界的远见卓识，并且识别出被多数人忽视的需求以及找出令人叹为观止的问题的解决方案。
大多数作者都面临着种种限制——包括物理环境，可用资源，或者特殊的需求定义——这些限制通常会使我们很难想象出解决方案。
而其他一些作者则是在已经存在解决方案的领域中重新研究，并且提出新的观点以及更好地实现某个功能。
本书的所有作者都从他们的项目中获得了一些经验。
不过在阅读完本书后，我们同样可以总结出一些更广泛的经验。
首先，在可靠和真实的规则能够真正应用之前，需要进行多次尝试。
因为，人们在维护稳定性、可靠性以及其他软件工程要求的标准时经常会遇到重重困难。
在这种情况下，我们通常没有必要抛弃支持这种承诺的原则。
有时候，从另一个角度来思考问题或许能够揭示一种新的方向，从而使我们在满足需求的同时无需牺牲那些好的技术。
另一方面，在有些章节中强调了这条古老的原则：在打破原则之前，人们必须首先了解这个规则。
有些作者在获得一种不同的解决方案之前积累了数十年的经验——而正是这些经验给了他们自信，从而以创造性的方式打破规则。
此外，书中的一些经验还提倡跨学科研究。
许多作者都是在新的领域中进行研究并在黑暗中不断探索。
在这种情况下，全新的创造力和个人智慧将起到重要的作用。
最后，我们从书中学到的漂亮的解决方案并不会持续很长时间。
在新的环境中总会要求新的解决方式。
因此，如果阅读了本书并且认为，“无法在自己的任何一个项目上使用这些作者的解决方案”，那么也不要担心——这些作者在做下一个项目的时候，也会使用不同的解决方案。
我在这本书上全身心地工作了两个月，以帮助作者完善他们的主题和更好地表达他们的观点。
阅读这些天才发明家的文章的确令人鼓舞甚至是令人情绪高涨的。
它给了我尝试新鲜事物的冲动，我希望读者在阅读本书时也能有同样的感受。
作者简介John Bentley是美国Avaya实验室的一位计算机科学家。
他的研究领域包括编程技术、算法设计以及软件工具与界面设计。
他已编写了数本关于编程的书籍，还撰写了大量的文章，主题涉及从算法理论到软件工程的各个方向。
他于1974年在斯坦福大学获得学士学位，并于1974年获得硕士学位以及于1976年在北卡罗来纳大学获得博士学位，随后在卡耐基-梅隆大学任教6年，教授计算机科学。
1982年他加入贝尔实验室，并于2001年离开贝尔实验室并加入Avaya实验室。
他曾是西点军校和普林斯顿大学的访问教授、曾经参与开发过软件工具、电话交换机、电话以及网络服务。
Tim Bray于1987-1989年间在加拿大的安大略省滑铁卢大学负责牛津英语词典项目，1989年与他人联合创建了Open Text公司，在1995年启动了最早的公共网页搜索引擎之一，在1996至1999年间与他人共同发明了XML 1.0并合作编写了《Namespaces in XML》规范，在1999年他创建了Antarctica Systems公司，并于2002-2004年被Tim Berners-Lee任命在W3C技术架构组中工作。
目前，他在Sun Microsystems公司Web Technologies部门任主管，他有一个很受欢迎的博客，并且参与主持IETF AtomPub工作组。
Bryan Cantrill是Sun Microsystems公司的一位杰出的工程师，在他的职业生涯中主要从事Solaris内核的开发。

最近他与同事Mike Shapiro和Adam Leventhal一起设计并实现了DTrace，这是一个用于产品系统动态控制的工具，获得了《华尔街日报》2006年度的最高创新奖。
Douglas Crockford毕业于公立学校。
他是一位登记选民，拥有自己的汽车。
他曾开发过办公自动化系统。
他曾在Atari公司从事过游戏和音乐研究。
他曾是Lucasfilm有限公司技术部门的主管，以及Paramount公司New Media部门的主管。
他创建了Electric Communities公司并且担任CEO。
他还是State 软件公司的创建者和CTO，正是在这个公司中他发明了JSON数据格式。
他现在是Yahoo!公司的一位架构师。
Rogerio Atem de Carvalho是巴西校园技术教育联合中心（Federal Center for Technological Education of Campos，CEFET Campos）的一位教师兼研究人员。
他在奥地利的维也纳获得了2006年度IFIP杰出学术领导奖（Distinguished Academic Leadership Award），以表彰他在免费/开源企业资源计划（ERP）上所做的研究工作。
他的研究领域还包括决策支持系统和软件工程。
Jeff Dean于1999年加入Google，目前是Google系统架构小组的成员。
他在Google主要负责开发Google的网页抓取、索引、查询服务以及广告系统等，他对搜索质量实现了多次改进，并实现了Google分布式计算架构的多个部分。
在加入Google之前，他工作于DEC/Compaq的Western实验室，主要从事软件分析工具、微处理器架构以及信息检索等方面的研究。
他于1996年在华盛顿大学获得了博士学位，与Craig Chambers一起从事面向对象语言的编译器优化技术方面的研究。
在毕业之前，他还在世界卫生组织的艾滋病全球规划署工作过。
Jack Dongarra于1972年在芝加哥大学获得数学学士学位，并于1973年在伊利诺理工大学获得计算机科学硕士学位，又于1980年在新墨西哥大学获得应用数学博士学位。
他在美国阿贡国家实验室（Argonne National Laboratory）一直工作到1989年，并成为了一名著名科学家。
他现在被任命为田纳西大学计算机科学系的计算机科学杰出教授。
他是美国橡树岭国家实验室（Oak Ridge National Laboratory，ORNL）计算机科学与数学部的杰出的研究人员，曼彻斯特大学计算机科学与数学学院的Turing Fellow，美国莱斯大学计算机科学系的副教授。
他的研究领域包括线性代数中的数值算法，并行计算，高级计算机架构的应用，程序设计方法学以及用于并行计算机的工具。
他的研究工作包括开发、测试高质量的数学软件以及整理相关文档。
他在以下开源软件包和系统的设计及实现上做出了贡献：ISPACK, LINPACK, the BLAS, LAPACK, ScaLAPACK, Netlib, PVM, MPI, NetSolve, Top500, ATLAS, 和 PAPI。
他公开发表了大约200篇文章、论文、报告以及技术备忘录，还参与编写了数本著作。
他于2004年获得了IEEE Sid Fernbach奖，以表彰他在高性能计算机的应用中使用了创新的方法。
他不仅是AAAS，ACM和IEEE的成员，还是美国工程院的院士。
R. Kent Dybvig是印第安纳大学计算机科学系的一位教授。
在印第安纳大学任教两年之后，他于1987年在北卡罗来纳大学获得了博士学位。
他在设计和实现编程语言的研究上做出了重要的贡献，包括控制运算符、句法抽象、程序分析、编译器优化、寄存器分配、多线程以及自动存储管理等。
在1984年，他创建了Chez Scheme软件并一直是主要的开发人员。
Chez Scheme的特点在于快速的编译时间、可靠性以及能够高效地运行内存需求巨大的复杂程序，它已经被用于构建企业集成、网页服务、虚拟现实、机器人药品抽检、电路设计以及其他的商业系统。
它还可以用于各种层次的计算机教育以及许多其他领域中的研究。

Dybvig是《The Scheme Programming Language, Third Edition》(MIT Press出版社)一书的作者，以及即将发布的“ Revised6 Report on Scheme ”文档的编辑。
Michael Feathers是Object Mentor 公司的顾问。
在过去七年间，他一直活跃于Agile社群，他的工作主要是与世界各地不同的团队合作，培训以及指导。
在加入Object Mentor公司之前，Michael设计过一种编程语言，并为这种语言写了一个编译器。
他还设计了一个庞大的多平台类库以及用于控制的框架。
Michael开发了CppUnit，也就是最初把JUnit移植到C++；以及FitCpp，也就是把FIT移植到C++。
在2005年，Michael编写了《Working Effectively with Legacy Code》（Prentice Hall出版社）一书。
在与各个团队合作的间隙，他的大多数时间都花在研究大型代码库中的设计修改方式方面。
1995年，Karl Fogel和Jim Blandy一起创建了Cyclic软件公司，这是第一个提供商业CVS支持的公司。
1997年，Karl增加了对CVS匿名只读存储仓库访问的支持，这样就可以更方便地访问开源项目中的开发代码。
1999年，他工作于CollabNet公司，主要从事管理Subversion的创建和开发工作，这是CollabNet公司和一群开源志愿者们从头开始编写的开源版本控制系统。
2005年，他编写了《Producing Open Source Software: How to Run a Successful Free Software Project》(O'Reilly出版社;在http://producingoss.com上有联机版本) 一书。
2006年，他在Google担任了短期的开源技术专家之后离开Google并成为了Question-Copyright.org网站的全职编辑。
他目前仍然参与了多个开源项目，包括Subversion和GNU Emacs。
Sanjay Ghemawat是一位Google Fellow，工作于Google的系统架构小组。
他设计并实现了分布式的存储系统，文本索引系统，性能分析工具，一种数据表示语言，一个RPC系统，一个malloc函数实现以及许多其他的库。
在加入Google之前，他是DEC系统研究中心的一位研究人员，主要从事系统性能分析和优化Java编译器的工作，他还实现了一个Java虚拟机。
他于1995年在麻省理工大学获得博士学位，研究领域为面向对象数据库的实现。
Ashish Gulhati是互联网隐私服务Neomailbox的首席开发员，以及Cryptonite的开发员，这是一个支持OpenPGP协议的安全网页邮件系统。
他有着15年的商业软件开发经验，是印度最早的数字版权活动家之一和F/OSS程序员，他编写了大量的开源Perl模块，这些模块可以从CPAN上下载。
在1993～1994年间，他在《PC Quest》和《DataQuest》等杂志上发表了大量文章，这是在印度主流计算机刊物中最早向读者介绍自由软件，GNU/ Linux，Web和Internet的文章，在这些文章发表多年以后，印度才拥有了商业的互联网访问，这些文章还构成了PC Quest Linux Initiative活动的重要组成部分，这个活动促使自1995年以来，在印度分发了一百万份Linux光盘。
在获得了一组可穿戴的计算机后，他很快地成为了一个电子人。
Elliotte Rusty Harold是新奥尔良人，他会定期返回新奥尔良去吃一大碗海鲜干波汤(Gumbo)。
不过，他目前住在布鲁克林附近的Prospect Heights，和他生活在一起还有他的妻子Beth，狗Shayna，和两只猫Charm（以夸克命名）和Marjorie（以他的岳母命名）。
他是纽约科技大学的一位副教授，主要讲授Java、XML以及面向对象编程。
他的Cafe au Lait网站(http://www.cafeaulait.org)是互联网上最流行的独立Java网站之一；他的另一个网站Cafe con Leche (http://www.cafeconleche.org)则成为了最流行XML站点之一。
他编写的书籍包括《Java I/O》，《Java Network Programming》和《XML in a Nutshell》（这三本书都由O'Reilly出版社出版），以及XML Bible (Wiley出版社)。
他目前的研究领域包括用Java来处理XML的XOM库、Jaxen XPath引擎以及Amateur媒体播放器。
Brian Hayes为《American Scientist》杂志编写计算机专栏，他还拥有一个博客http://bit-player.org。
过去，他还为《Scientific American》、《Computer Language》、以及《The Sciences》等杂志编写过类似的专栏。

他编写的《Infrastructure: A Field Guide to the Industrial Landscape》(Norton出版社)一书于2005年发行。
Simon Peyton Jones，硕士，于1980年毕业于剑桥大学三一学院。
在工作两年后，他在伦敦大学学院担任了7年的讲师，然后在格拉斯哥大学担任了9年的教授，后来于1998年加入微软研究中心。
他的研究领域包括函数式编程语言及其实现和应用。
他领导了一系列的研究项目，主要研究用于单处理器机器和并行机的高质量函数式语言系统的设计和实现。
他是函数式语言Haskell的主要设计者，此外他还是被广泛应用的Glasgow Haskell编译器（GHC）首席设计师。
他还编写了两本关于函数式语言实现的教科书。
Jim Kent是加利福尼亚大学圣克鲁兹分校基因信息小组（Genome Bioinformatics Group）的一位研究学家。
Jim从1983年起就开始编程。
在职业生涯的前半段，他主要从事绘画和动画软件的开发，他开发了Aegis Animator、 Cyber Paint以及Autodesk Animator等获奖软件。
1996年，由于厌倦了基于Windows API的开发工作，他决定在生物学上追求他的兴趣，并于2002年获得了博士学位。
在研究生期间，他编写GigAssembler——这个程序计算出了第一批人类基因组——比Celera公司发布的第一批基因组提前了一天，从而使得这批基因组成为免费的专利并且避免了其他的法律问题。
Jim发表了40余篇科学论文。
他目前的研究工作主要是编写程序，数据库和网站以帮助科学家分析和了解基因组。
Brian Kernighan于1964年在多伦多大学获得学士学位，并于1969年在普林斯顿大学获得电子工程博士学位。
他在贝尔实验室的计算科学研究中心一直工作到2000年，目前就职于普林斯顿大学的计算机科学系。
他编写了8本著作以及大量的技术论文，并拥有4项专利。
他的研究领域包括编程语言、工具、为非专业用户设计易用的计算机操作界面等。
他还致力于非技术读者的技术教育工作。
Adam Kolawa是Parasoft公司的创建者之一和CEO，这家公司是自动错误预防（Automated Error Prevention，AEP）解决方案的领先提供商。
Kolawa有着多年在各种软件开发流程中的经验，这使得他对高科技企业有着独特的视野，以及成功辨识技术潮流的非凡能力。
因此，他策划了几个成功商业软件产品的开发过程来满足在提高软件质量中不断增长的工业需求——经常在这种潮流被广泛接受之前。
Kolawa参与编写了《Bulletproofing Web Applications》(Hungry Minds出版社)一书，他还撰写了100余篇评论和技术文章，发表在《The Wall Street Journal》、《CIO》、《Computerworld》、《Dr. Dobb's Journal》以及《IEEE Computer》等期刊上。
此外，他还撰写了大量关于物理学和并行处理方面的科学论文。
他现在的签约媒体包括CNN、 CNBC、 BBC和NPR。
Kolawa拥有加利福尼亚理工大学理论物理博士学位，并拥有10项专利发明。
2001年，Kolawa获得了软件类别的Los Angeles Ernst & Young's Entrepreneur of the Year奖项。
Greg Kroah-Hartman是目前Linux内核的维护人员，负责多个驱动程序子系统以及驱动程序内核、sysfs、 kobject、 kref和debugfs等代码。
他还为启动linux-hotplug 和 udev等项目提供了帮助，是内核稳定维护团队中的重要人员。
他编写了《Linux Kernel in a Nutshell》(O'Reilly出版社)，并参与编写了《Linux Device Drivers, Third Edition》(O'Reilly出版社)。
Andrew Kuchling有着11年的软件工程师经验，他是Python开发群体中的长期成员。
他的一些与Python相关的工作包括编写和维护数个标准的库模块，编写一系列的“ What's new in

Python 2.x ”文章以及其他一些文档，策划了2006年和2007年的PyCon会议，并是Python软件基金会的主管。
Andrew于1995年毕业于麦吉尔大学并获得计算机科学学士学位。
他的个人网页是http://www.amk.ca。
Piotr Luszczek毕业于波兰克拉科夫矿业与冶金大学，并获得硕士学位，他的研究领域是并行的核外（out-of-core）库。
他将稠密矩阵计算核应用于稀疏矩阵直接求解算法和迭代数值线性几何算法中的创新研究使他获得了博士学位。
他把这种思想用来开发使用核外技术容错库。
目前，他是田纳西大学诺克斯维尔分校的一位研究教授。
他的研究工作包括大型超级计算机安装的标准化评价。
他开发了一个自适应的软件库，能够自动选择最优的算法来有效地利用现有硬件以及有选择地处理输入数据。
他还感兴趣于高性能编程语言的设计和实现。
Ronald Mak是高级计算机科学研究所（Research Institute for Advanced Computer Science）的一位资深科学家，在NASA Ames研究中心工作时，他是协同信息系统（Collaborative Information Portal，CIP）的架构师和首席开发人员。
在漫步者登录火星之后，他分别在JPL和Ames对探测任务提供支持。
然后，他获得了加利福尼亚大学圣克鲁兹分校的学术任命，并且他再次与NASA签约，这次的工作是设计帮助宇航员返回月球的企业软件。
Ron是Willard & Lowe Systems(http://www.willardlowe.com)公司的创建人之一和CTO，这是一个针对企业信息管理系统的咨询公司。
他编写了数本关于计算机软件的书籍，他在斯坦福大学分别获得了数学科学学位和计算机科学学位。
Yukihiro "Matz" Matsumoto是一位程序员，他是一位日本籍的开源倡导者，他发明了最近非常流行的Ruby语言。
他从1993年开始研发Ruby，这和Java语言一样久远。
现在他工作于日本Network Applied Communication Laboratory（NaCl，网址为netlab.jp）公司，该公司从1997年起开始赞助Ruby的开发。
因为他的真实姓名太长而难以记住，并且对于非日本的演讲者来说难以发音，因此在网上他使用了昵称Matz。
Arun Mehta是一位电子工程师和计算机科学家，他曾在印度、美国和德国进行过研究和教学工作。
他是印度早期计算机活动家，他努力实现了一些方便消费者（consumer-friendly）的政策，以帮助把现代通信延伸到偏远地区和贫困地区。
他目前的研究领域包括农村无限通信以及帮助残疾用户的技术。
他是印度哈里亚纳邦Radaur地区JMIT大学计算机工程系的教授和主任。
他的网址包括http://india-gii.org, http://radiophony.com和 http://holisticit.com。
Rafael Manhaes Monnerat是CEFET CAMPOS的一位IT分析家，以及Nexedi SARL的海外顾问。
他的研究领域包括免费/开源系统、ERP以及最新的编程语言。
Travis E. Oliphant于1995年在美国杨百翰大学获得电子与计算机工程学士学位和数学学士学位，并于1996年在本校获得电子与计算机工程硕士学位。
他于2001年在明尼苏达罗切斯特的梅奥研究生院获得了生物医学工程博士学位。
他是Python语言中科学计算库SciPy和NumPy的主要编写者。
他的研究领域包括显微阻抗成像，异构领域中的MRI重构以及生物医学逆问题。
他目前是杨百翰大学电子与计算机工程的副教授。
Andy Oram是O'Reilly Media的编辑。
他从1992年开始就在这家公司工作，Andy目前主要关注自由软件和开源技术。
他在O'Reilly的工作成果包括第一批Linux系列丛书以及2001年的P2P系列丛书。

他的编程技术和系统管理技术大多都是自学的。
Andy还是Computer Professionals for Social Responsibility协会的成员并且经常在O'Reilly Network(http://oreillynet.com)和其他一些刊物上撰写文章，这些文章的主题包括互联网上的政策问题，以及影响技术创新的潮流及其对社会的影响。
他的网址为http://www.praxagora.com/andyo。
William R. Otte是田纳西范德堡大学电子工程与计算机系（EECS）的一位博士研究生。
他的研究领域是分布式实时嵌入（DRE）系统的中间件，目前从事CORBA组件的部署和配置引擎（DAnCE）开发工作。
这个工作主要研究运行时规划技术，基于组件的应用程序的适应性，以及对应用程序服务质量和容错需求的规范与实施。
在攻读研究生之前，William于2005年在范德堡大学计算机系毕业并获得学士学位，之后在软件集成系统学院（ISIS）工作了一年。
Andrew Patzer是威斯康星大学医学院生物信息系的主管。
过去15年Andrew是一位软件开发人员并且编写了许多文章和书籍，包括《Professional Java Server Programming》（Peer Information公司）和《JSP Examples and Best Practices》(Apress出版社)。
Andrew目前的研究领域为生物信息领域，利用像Groovy这样的动态语言来发掘大量有效的生物数据并帮助科学研究人员进行分析。
Charles Petzold是一位自由作家，主要研究领域为Windows应用程序编程。
他是《Programming Windows》(Microsoft Press出版社)的作者，1988年至1999年之间共出版了五版，教育了整整一代程序员的Windows API编程技术。
他最新的书籍包括《Applications = Code + Markup: A Guide to the Microsoft Windows Presentation Foundation》(Microsoft Press出版社)，以及《Code: The Hidden Language of Computer Hardware and Software》(Microsoft Press出版社)，在这本书中他对数字技术进行了独特的研究。
他的网址是http://www.charlespetzold.com。
T. V. Raman的研究领域包括网页技术和听觉用户界面。
在20世纪90年代初，在他的博士论文中介绍了音频格式的概念，叫作AsTeR：Audio System For Technical Readings（技术读物语音系统），这是一个为技术文档生成高质量听觉表示的系统。
Emacspeak则将这些思想应用到更广泛的计算机用户界面领域。
Raman现在是Google的一位研究人员，主要研究Web应用程序。
Alberto Savoia是Agitar软件公司的创建人之一和CTO。
在创建Agitar之前，他是Google的高级工程主管；在这之前，他还是Sun Microsystems实验室软件研究中心的主管。
Alberto的主要研究领域是软件开发技术——尤其是那些帮助程序员在设计和开发阶段进行测试和代码验证的工具和技术。
Douglas C. Schmidt是田纳西范德堡大学电子工程与计算机（EECS）系的一位教授，计算机科学与工程系的副主任，以及软件集成系统学院（ISIS）的高级研究人员。
他是分布式计算模式和中间件框架方面的专家，并且已经发表了超过350篇的技术论文和9本书籍，内容涉及的主题很广，包括高性能通信软件系统，高速网络协议并行处理，实时分布式对象计算，并发与分布式系统的面向对象模式，以及模型驱动的开发工具。
在他的学术研究之外，Dr. Schmidt还是PrismTechnologies公司的CTO，并且在领导开发应用广泛开源的中间件平台上有着15年的经验，在这些平台上包含了丰富的组件以及实现高性能分布式系统中核心模式的领域特定语言。
Dr. Schmidt于1994年于加利福尼亚大学欧文分校获得计算机科学博士学位。
Christopher Seiwald编写了Perforce（一种软件配置管理系统）、Jam（一种构建工具）和“漂亮代码的七个要素”（本书的第32章，变动的代码，正是从这篇文章中提取出了有价值的思想）。
在创建Perforce之前，他在Ingres公司管理网络开发小组，他花了数年时间来使得异步网络代码看上去很漂亮。

现在他是Perforce软件公司的CEO，并且仍然从事编码工作。
Diomidis Spinellis是希腊雅典经济与商业大学管理科学与技术系的副教授。
他的研究领域包括软件工程工具，编程语言和计算机安全。
他在伦敦帝国理工大学获得了软件工程硕士学位和计算机科学博士学位。
他发表了超过100篇的技术论文，所涉及的领域包括软件工程，信息安全以及普适计算。
他还编写了两本开源方面的书籍：《Code Reading》(获得2004年度Software Development Productivity奖)和《Code Quality》(这两本书都由Addison-Wesley出版社出版)。
他是IEEE Software编辑委员会的成员，主编“Tools of the Trade”专栏。
Diomidis是一位FreeBSD提交者（Committer），并且编写了许多开源软件包、软件库以及工具。
Lincoln Stein是一位硕士/博士，他的研究领域为生物信息数据的集成与虚拟化。
在从哈佛大学医科学院毕业后，他在麻省理工大学Whitehead基因研究所工作，开发用于老鼠和人类的基因图谱数据库。
他在冷泉港实验室开发了各种基因数据库，包括WormBase，线虫基因数据库；Gramene，用于水稻和其他单子叶植物的比较基因映射数据库；国际Hap-Map项目数据库；以及人类基因基础数据库Reactome。
Lincoln还编写了《books How to Set Up and Maintain a Web Site》(Addison-Wesley出版社)、《Network Programming in Perl》(Addison-Wesley出版社)、《Official Guide to Programming with CGI.pm》(Wiley出版社)以及《Writing Apache Modules with Perl and C》(O'Reilly出版社)等书籍。
Nevin Thompson把Yukihiro Matsumoto编写的第29章内容，把代码当作文章，从日文翻译到英文。
他的客户包括日本最大的电视网络，以及Technorati Japan公司和Creative Commons组织。
Henry S. Warren, Jr.在IBM工作了45年，他历经了从IBM 704到PowerPC的发展过程。
他参与过多个军方指挥与控制系统的开发工作，在纽约大学Jack Schwartz教授指导下从事SETL项目。
从1973年起，他在IBM研究部门工作，主要方向为编译器和计算机架构。
Hank目前正在参与Blue Gene Petaflop超级计算机项目。
他在纽约大学克朗数学研究所获得了计算机博士学位。
他是《Hacker's Delight》(Addison-Wesley出版社)一书的作者。
Laura Wingerd多年Sybase和Ingres的数据库产品开发工作形成了她早期对软件配置管理的观点。
她在Perforce软件公司创建之初就加盟了这家公司，并且从她给Perforce客户的建议中获得了大量的SCM经验。
她编写了《Practical Perforce》(O'Reilly出版社)一书以及许多与SCM相关的白皮书。
她在Google的技术演讲The Flow of Change中首次露面。
Laura现在是Perforce 软件公司产品技术部的副主管，主要负责推动合理的SCM流程以及研究新的并且更好的Perforce使用方式。
Greg Wilson在爱丁堡大学获得了计算机科学博士学位，他的研究领域包括高性能科学计算，数据虚拟化以及计算机安全。
他现在是多伦多大学计算机科学系的一位副教授，并且是《Dr. Dobb's Journal》杂志的特约编辑。
Andreas Zeller于1991年毕业于德国达姆斯塔特理工大学，并于1997年在不伦瑞克理工大学获得计算机科学博士学位。
2001年以来，他一直在德国萨尔兰登大学的计算机科学系担任教授。
Zeller主要研究大型程序以及它们的发展历史，他开发了大量的方法来分析在开源软件以及IBM、Microsoft、SAP以及其他公司的商业软件中失败的原因。
他编写的《Why Programs Fail: A Guide to Systematic Debugging》(Morgan Kaufmann出版社)获得了《Software Development Magazine》杂志2006年度的Productivity大奖。

媒体关注与评论

<<代码之美>>

编辑推荐

38位大师级的程序员，一步步讲解他们的项目架构，开发时的种种折中考虑（tradeoffs）以及何时必须打破常规，寻求突破。
全球38位顶尖高手、众多语言之父经典之作《代码之美》，9月22s日全国首映

版权说明