资深软件开发专家Josh Carter 20余年编程生涯心得体会，从程序员成长视角，系统总结和阐述了专业程序员在编程技能和方法、编程工具、自我管理、团队协作、组织架构、工作态度和原则、自我学习和持续改善等方面应该掌握的33个技巧。

华章科技

The Pragmatic Programmers

华章程序员书库

New Programmer's Survival Manual
Navigate Your Workplace, Cube Farm, or Startup

程序员修炼之道

专业程序员必知的33个技巧

（美） Josh Carter 著
胡键 译

机械工业出版社
China Machine Press

华章程序员书库

New Programmer's Survival Manual
Navigate Your Workplace, Cube Farm, or Startup

程序员修炼之道

专业程序员必知的33个技巧

（美） Josh Carter 著
胡键 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

程序员修炼之道：专业程序员必知的33个技巧 /（美）卡特（Carter, J.）著；胡键译 . —北京：机械工业出版社，2013.1
（华章程序员书库）
书名原文：New Programmer's Survival Manual: Navigate Your Workplace, Cube Farm, or Startup

ISBN 978-7-111-41164-2

Ⅰ . 程… Ⅱ . ① 卡… ② 胡… Ⅲ . 程序设计 – 基本知识 Ⅳ . TP311.1

中国版本图书馆 CIP 数据核字（2013）第 006745 号

本书版权登记号：图字：01-2012-6853

这是每一位致力于成为专业程序员的软件开发新手都应该阅读的一本书。它是资深软件开发专家Josh Carter 20余年编程生涯的心得体会，从程序员成长的视角，系统总结和阐述了专业程序员在专业技能、编程工具、自我管理、团队协作、工作态度以及需要采取的行动等方面应该掌握的33个非常重要且实用的技巧。作者以自己以及身边的同事积累下来的经验、犯过的错误为素材，旨在为新人们引路，让他们在能力修炼的过程中少走弯路！

全书分为四个部分：第一部分（技巧1~14），从编程技能和工具使用两个方面总结了14个技巧，包含如何正确地书写代码、测试驱动设计、管理代码复杂度、改善遗留代码、代码评审、开发环境优化、自动化等；第二部分（技巧15~24），从自我管理和团队协作两个方面总结了10个技巧，包括如何树立自我形象、压力管理、建立良好人脉和高效会议等；第三部分（技巧25~30），介绍了典型高科技公司的组织结构以及你在整个公司中的位置，并且阐述了薪酬分配的问题；第四部分（技巧31~33），介绍了在日常工作中如何持续改善自己的工作和学习状态。

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）
责任编辑：秦 健
三河市杨庄长鸣印刷装订厂印刷
2013 年 1 月第 1 版第 1 次印刷
186mm×240mm • 13.25 印张
标准书号：ISBN 978-7-111-41164-2
定价：49.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

译 者 序

说实话，在翻译完《The Cloud at Your Service》[⊖]之后，我本打算暂时不再接手翻译工作了。一方面是因为工作越来越忙，另一方面则是因为翻译实在是一件费神费力的事情。可当机械工业出版社的编辑把这本书摆在我面前时，我还是背弃了自己想要休息的想法。

我注意这本书其实有一段时间了，因为这是一本独特的书。虽然面向程序员，却并非一本单纯的技术书籍。尽管英文书名包含 New Programmer，但对于入行多年的“老兵”同样能做到开卷有益。事实上，在翻译本书的过程中，我本人也从中获益匪浅。

对于刚踏入社会工作的毕业生，要完成从学生到社会人的转变并非易事。尤其是那些从事编程工作的社会新鲜人，有许多的观念和事情需要去转变和学习。比如：你以前可能孤身一人开发完成整个应用，现在则可能要跟他人合作；你之前的代码可能是自己分目录保存不同副本，如今可能要用到类似 Git 这样的版本控制软件；你之前的代码可能是自己手动测试，而新公司可能要求你写自动化的单元测试……除了这些纯技术上的变化外，你还会面临一些诸如绩效考核、配合市场人员宣传、职业生涯这样的新事物。

所有以上内容，你都会在本书中找到相应的内容和建议。虽然读一本书不会让你马上转变身份，但起码会给你提供帮助，让你感觉不再孤单。说句心里话，要是当年我有一本这样的书该有多好！

虽然时光无法倒流，但现在也为时未晚。对于在踏入编程大门初期错过了本书的朋友们，同样能在本书中找到适合自己的内容。你是否每到绩效考核时就头痛？你是否觉得长时间坐在办公室内编程损害了你的健康？你是否讨厌开会，认为它们纯粹是浪费时间？你是否关心所在公司的营业情况？你有没有想过换个工作？……打开这本书，读一读别人的想法和做法。

除了本书的内容外，我接下这本书的翻译其实还有一个私心：希望我的同行妻子也能从中汲取些营养。于是，在“初译—审校—润色—审校”的过程中，她最终承认这是一本好

⊖ 本书中文版《云计算揭秘——企业实施云计算的核心问题》已由机械工业出版社引进出版，ISBN: 978-7-111-38494-6。——编者注

书，决定要将从书中学到的经验用到实际中——这是“润物细无声”的典型案例。

最后，我要感谢参与本书的所有的机械工业出版社的编辑，让我有机会负责本书的翻译工作，而且有一个非常愉快的合作体验；同时，我还要感谢参与本书审校工作的朋友：朱晓弟、仵建锋和焦斌。当然也不能遗漏老婆，不仅是因为你的审校次数最多、最深入，而且还因为你对于我生活的支持和帮助！

胡　键

于西安

前　言

今天是上班的第一天。你拿到了编程执照，找到了工作，坐在你的工作站旁……下面该干什么？在你面前，一座新的丛林正等着你：

- 按行业规模编程，其中的代码库规模以上千（或几十万）行代码来衡量。你怎样才能快速入门，开始作出贡献？
- 遨游在除了程序员之外还有许许多多其他角色的组织内。当要了解产品特性时，你向谁请教？
- 每年都有所成就。当绩效考评潜伏在地平线上时，你知道老板的目标吗？你知道自己将如何被判定吗？

还有很多很多。你的编程技能只是工作第一年里要用到的技能的一部分。

我们中的幸运儿会有识途老马充当向导。本书则是一位虚拟向导，它将为你指明方向，指出前方的高山和峡谷，同样也将让你避免跌入令人讨厌的陷阱。

我的经历

你或许能从我在 1995 年上大学时的情景中找到一些与自己经历相似之处：我一开始走的是传统老路，一名杜克大学计算机科学与电子工程系的学生。我曾找过我的导师，询问哪些课程最有利于我未来求职。他是个聪明的家伙——一名罗德学者[⊖]和这间工程学校冉冉升起的新星——他的回答是："我不知道。我从未在行业里工作过一天。"

我大失所望。我想构建真实、有人买的产品，而不是写研究论文。因此，那个夏天我设法加入了硅谷一家方兴未艾的创业公司：General Magic。它是由当初创造了 Macintosh 计算机的同一拨人（Andy Hertzfeld 和 Bill Atkinson）创建的。我的同事包括来自苹果公司 System 7（操作系统）团队的一些顶级开发者和后来创建了 eBay 的那个哥们儿。

⊖ 罗德奖学金得主的称号。罗德奖学金由塞西尔·罗德斯于 1902 设立，已有超过百年历史，它是世界级的奖学金，有"全球本科生诺贝尔奖"之称的美誉。（摘自维基百科）——译者注

我在两个月的实习期内学到的编程知识比我在学校里两年学到的还要多。我给杜克大学打了个电话，说我不打算回学校了。就这样，我在行业里的狂野冒险开始了。

现在说说你

本书的读者可大致划为如下几类：

- 选修计算机科学和有这样疑问的大学生和将要毕业的学生："现实世界里的编程是这个样子吗？"（简单说：不是。）
- 具有其他背景，因为爱好或副业而涉足编程，现在想将其作为全职工作的职业人士。
- 正在考虑编程行当，但想找些书中和课堂上没有教过的东西的其他人。

不论你属于哪种类型，你现在的情况是：到了靠编码为生的时候。就代码部分而已，市面上以之为主题的书可谓汗牛充栋。但讨论跟这个工作相关的其他方方面面的书籍，就不见得有那么多了——这正是本书的初衷。

对于转行的从业者，有些章节可能对你没多大用处——假如你具有市场营销的背景，那就用不着我来告诉你市场营销究竟为何物。但你还是可以从工程部门的运作方式以及代码从概念到发布的演变过程的相关内容中获益。

本书组织结构

本书以技巧的形式写就，每个技巧用寥寥数页说明某个主题，有些技巧可能稍长。相关的技巧组织在一起形成章，但阅读它们的顺序可以由你来定。若想了解全景图，那就一页页从头读到尾。但可随意来回翻阅——当技巧需要彼此引用时，会在文中明确指出。

一开始的讨论跟代码密切关联：第 1 章"编程生产"从你擅长的编程出发，就如何让代码随时可以用于生产环境提供了指导。没有人想让充满 Bug 的代码面市，但在行业规模的项目上确保代码正确并经过良好测试却是不小的挑战。

紧随而至的第 2 章"整理工具"将帮你改善工作流程。你需要跟他人协调，自动化构建，并在工作过程中学习新技术。此外，你还要输入"成吨"的代码。值得提前对工具有所投入。

随后，我们会进入事情更麻烦的一面。陪伴你度过此生的那位经理就是你自己，第 3 章"自我管理"让你开始注意诸如压力管理和工作绩效等这类问题。

没有程序员是孤立的，第 4 章“团队协作”关注与他人的合作。不要低估人员技能——没错，擅长使用计算机是雇佣你的原因，但编程行业是一项团体活动。

接着，我们将了解宏观景象。第 5 章“走进公司”考虑了典型高科技公司的所有组成部分，以及你在整个公司中的位置。它最终试图回答，“这些家伙成天都在忙什么？”

软件企业充满风险。第 6 章“留意你的企业”谈论谁以及为何要支付你薪水，软件项目的生命周期，以及你的日常编程工作如何随那个生命周期发生改变。

最后，第 7 章“改善”将放眼未来。日语“改善”（kaizen）是一种持续改进的哲学，在我们分手之前，我希望看到你走在那条道路上。

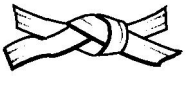
本书约定

我经常在包含示例代码的技巧里使用 Ruby 编程语言，我选择 Ruby 仅仅是因为它简洁易读。若不懂 Ruby 也别担心，代码的意图应该一目了然。这些例子意在阐述适用于任何编程语言里的高层原则。

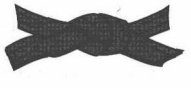
贯穿全书，你会看到题为“行业观点”的栏目。这些都是来自行业专家的声音，这些人是早于你走上这条道路的程序员和经理。每位贡献者都有超过 10 年的经验，因此请慎重考虑他们的建议。

从白带到黑带（再回到白带）

在整本书里，我会在你需要应用某条技巧的时候使用武术带来表示。带子颜色背后的故事要比武术本身更有意义。学生入门从白带开始，代表纯洁。同理，白带技巧适用于非常初级的阶段。



经过几年修行，带子变或褐色了。褐带代表中间阶段，坦白地讲，这时的带子是有点脏。就本书而言，我期望褐带主题对于工作第 2 ～ 5 年之间的人有帮助。



Robert C. Martin、Rajesh Pillai、Antonio Gomes Rodrigues、Sam Rose、Brian Schau、Julian Schrittwieser、Tibor Simic、Jen Spinney、Stefan Turalski、Juho Vepsäläinen、Nick Watts 和 Chris Wright。他们用勤奋和彻底的评审大大提升了本书的质量。

从一开始，几位朋友和合著者就纵容我紧追不舍、一遍又一遍地征询他们的意见，他们包括 Jeb Bolding、Mark "The Red" Harlan、Scott Knaster、David Olson、Rich Rector 和 Zz Zimmerman。我真的很感激他们的容忍。

最后，额外的感谢要献给我的两个最大粉丝。我的女儿 Genevieve，恩赐了我无法陪她而用来写书的许许多多夜晚。还有我的妻子 Daria，不仅让我有时间写作，而且第一时间购买并阅读了本书的 beta 版本——从晚上 10 点开始一气读完，有点出人意料。本书是我坐在晚餐桌旁一直考虑的点子，因此她也贡献了自己的想法和观点。同时，在整个过程中，她也提供了支持和鼓励。

Daria 和 Genevieve，没有她们，我根本没法完成本书，衷心地感谢她们。

目　录

译者序
前言

第一部分　专业编程

第 1 章　编程生产 …… 2
技巧 1：敲打代码 …… 4
技巧 2：坚持正确 …… 9
技巧 3：测试驱动设计 …… 19
技巧 4：驯服复杂度 …… 25
技巧 5：优雅地失败 …… 33
技巧 6：确定风格 …… 39
技巧 7：改善遗留代码 …… 45
技巧 8：代码审核要早且多 …… 50
第 2 章　整理工具 …… 55
技巧 9：优化环境 …… 57
技巧 10：流畅表达 …… 64
技巧 11：熟悉平台 …… 71
技巧 12：自动让痛苦消失 …… 76
技巧 13：控制时间及时间线 …… 79
技巧 14：使用源码，卢克 …… 83

第二部分　人员技能

第 3 章　自我管理 …… 92
技巧 15：拜师 …… 93
技巧 16：树立自我形象 …… 97
技巧 17：增加曝光率 …… 100

技巧 18：表现卓越 …… 103
技巧 19：管理压力 …… 109
技巧 20：善待自己 …… 115
第 4 章　团队协作 …… 120
技巧 21：洞悉性格类型 …… 121
技巧 22：建立人脉 …… 126
技巧 23：合作 …… 129
技巧 24：高效会议 …… 133

第三部分　公司的世界

第 5 章　走进公司 …… 138
技巧 25：了解同事 …… 139
技巧 26：了解公司结构 …… 144
第 6 章　留意你的企业 …… 159
技巧 27：了解项目 …… 160
技巧 28：体会产品的生命周期 …… 166
技巧 29：站在公司角度思考 …… 176
技巧 30：识别公司反模式 …… 179

第四部分　放眼未来

第 7 章　改善 …… 184
技巧 31：端正态度 …… 186
技巧 32：学无止境 …… 189
技巧 33：自我定位 …… 193

参考文献 …… 197

第一部分

专业编程

技巧 1：敲打代码
技巧 2：坚持正确
技巧 3：测试驱动设计
技巧 4：驯服复杂度
技巧 5：优雅地失败
技巧 6：确定风格
技巧 7：改善遗留代码
技巧 8：代码审核要早且多
技巧 9：优化环境
技巧 10：流畅表达
技巧 11：熟悉平台
技巧 12：自动让痛苦消失
技巧 13：控制时间及时间线
技巧 14：使用源码，卢克

第①章

编程生产

将编程作为一种娱乐消遣，你会轻易忽略像边界情况处理、错误报告等这类事情，它们确实有点麻烦。可一旦你从事编程生产（更别提要养家糊口）你就不能走捷径。

编写生产质量级别的代码似乎是一个明摆着的目标，但计算机行业却费了不少时日才弄明白正确的实现之道。例如，Windows 95 曾经有个 Bug 会让操作系统在连续运行 49.7 天之后挂起——但是该 Bug 花了 4 年时间才暴露，有 Bug 这件事本身并不特别让人觉得惊讶，时间之所以这么长是因为其他 Bug 在不到 49.7 天的时候就让 Windows 95 崩溃了。⊖

通往高质量代码的道路有两条，你可以二选一：一开始就内置质量，或者事后再敲打它。前者需要你在日复一日的编码中遵循众多戒律；后者则要求大量测试，到头来，在自以为完工之后，你会发现还有很多工作要做。

事后敲打（beat-it-in-afterward）是常见的工作方式，行业占统治地位的瀑布开发方法就是这样：规格说明、设计、构建、测试。测试是最后的步骤。产品来到测试部门，很快就崩溃了。于是，又回到工程部门，修复 Bug。接着，把另一版提交给测试部门，又由于其他原因崩溃。就这样，来来回回，许多月（甚至是数年）流逝。

本章大部分内容都聚焦于内置质量的技术，因为它是一种可以让你对自己的产品有信心、给产品添加新特性，并长年维护产品的构建方法。当然，构置生产质量级别的软件并不是一本书就可以完全覆盖的主题，而且它的范畴也要比测试大得多。不管怎样，本章仅限于讨论对改进代码质量可以起到立竿见影效果的那些事情：

- 深入具体实践之前，我们会从“技巧 1：敲打代码”开始，帮你建立正确的思维方式。

⊖ http://support.microsoft.com/kb/216641

- 接下来是“技巧 2：坚持正确”，我们将关注验证代码正确性的方法。
- 你还可以用另一种方法；在“技巧 3：测试驱动设计”中，我们尝试从测试出发，使用这些测试来驱动设计。
- 很快，你就会被巨大的代码库给弄得晕头转向。“技巧 4：驯服复杂度”尤其适合外表吓人的生产规模的软件项目。
- “技巧 5：优雅地失败”会把我们带离愉悦之路，在那里，你的代码需要应对意想不到的问题。
- 在事情才真正变得棘手的时候，我们会小憩一会儿：“技巧 6：确定风格”帮助你让代码保持美观，从长远来看，它对你的帮助超乎想象。
- 回到困难的部分。“技巧 7：改善遗留代码”处理你从前辈那里继承而来的代码。
- 最后，在“技巧 8：代码评审要早且多”中，你将和你的团队一起来保证你的代码随时可供部署。

关于这里没有谈及的内容

限于篇幅，还有其他一些对编程生产有促进的内容我并没有提及，而且在很多行业内都有你需要满足的领域相关的标准包括如下一些示例：

- 预防恶意代码、网络活动和其他安全性问题的防御性编程。
- 保护用户的数据不会因硬件和系统故障、软件 Bug 和安全破坏而受损。
- 部署并在面临巨大负载时软件性能的向外扩展。

……

向资深程序员求教：除了写出能工作的代码外——要一贯保持——还要做什么才能让你的代码合格？

技巧 1

敲打代码

[白带] 只要编写生产代码，你就要证明它经得起推敲。

你可能认为编写可靠代码是再明显不过的工作要求了。招工广告上不可能写："急聘：具备良好工作态度、团队合作精神和桌上足球技巧的程序员。有则更佳：会编写可靠的代码。"可有问题的程序还是有这么多，怎么回事？

在深入探讨保证代码质量的日常实践之前，让我们先讨论"编写可靠代码"的含义。它不仅仅是一份实践清单，它还是一种思维方式。在把产品交到客户手中之前，你必须敲打自己的代码和整个产品。

客户终究敲打你的产品，以一种你不曾预料到的方式使用它。他们用它的时间会很长，而且会在你没有测试过的环境里用它。你必须考虑的问题是：打算让客户发现多少 Bug？

你现在对代码敲打的次数越多，在交到客户手中之前，能清除掉的 Bug 就越多，留给客户的 Bug 就越少。

质量保证的形式

尽管本章大部分内容都关注于代码级的质量和单元测试，但品质保证却是一个要大得多的主题。让我们考虑一下产品需要经受的考验。

代码评审

保证代码质量最简单的方法就是让另一个程序员去读它。别出心裁的评审过程并没有必要，而且就连结对编程也算是一种形式的实时代码评审。团队将利用代码评审捕获 Bug，贯彻编程风格和标准，同时在团队成员间传播知识。我们将在"技巧 8：代码评审要早且多"中讨论代码评审。

单元测试

在你一个类接着一个类、一个方法接着一个方法地构建应用的业务逻辑时，验证代码的最佳方式就是单元测试。这种内部零件级的测试被设计用来对逻辑的各部分单独验证。我们将在“技巧2：坚持正确”和“技巧3：测试驱动设计”中讨论它们。

接受测试

单元测试立足于由内而外地审视产品，接受测试则被设计成模拟真实世界的用户，代表他们与系统交互。理想状况下，它们是自动执行的，而且以某种叙述式的风格书写出来。例如，某银行自动柜员机应用会有类似这样的接受故事：若我的活期存款为0，当我在ATM的“活期存款”中选择“取款”时，那么我应该看到“对不起，今天的晚餐吃泡面吧。”

它不像莎翁著作那样文采飞扬，但这些测试操练了整个系统：从用户界面一直到业务逻辑。无论它们是自动执行的，还是人工执行的，你的公司需要知道——在任何客户使用它之前——所有系统组件正在像预期的那样协调工作。

负载测试

负载测试将产品置于真实的压力条件下，然后度量它的响应。例如，某网站可能需要在数据库有100万条记录的条件下在100毫秒内展示指定页面。这些测试将揭示正确但不恰当的行为，如需要线性伸缩但却以指数级别伸缩的代码。[⊖]

定向探索测试

接受测试覆盖了产品的所有指定行为，它可能来自于产品需求文档或会议。但程序员通常还是有办法使之崩溃——总有些黑暗角落被规格说明疏忽掉。定向探索测试就是要将这些边界情况挖出来。

⊖ 例如，期望响应时间与负载之间的关系是线性关系，但实际却是指数关系。——译者注

“全系统”测试的范围究竟有多大？

我曾耗费数年编写工业机器人的控制软件。由于单元测试会模拟电动机的运动，我得以能够在工作站上测试业务逻辑。全系统测试当然需要运行在真正的机器人上。

就机器人来讲，好的事情是你能看到自己的代码能工作，不太好的事情是你能看到（和听到，有时甚至闻到）代码的失败。但更重要的是，机器人不是一个完美的环境。每个机器人都不同——它是上千个机电零件的组合体，每个零件多少都有些差异。因而，在多个机器人上测试非常关键。

这条经验对于更传统的系统同样适用：供应商的软件可能崩溃，网络会有延迟，硬盘会取出坏数据。公司的测试实验室应该模拟这些非理想环境，因为产品最终将会在客户手中遇到这些状况。

这种测试通常是人工执行的，可能是程序员自己，用于探索和发现问题。但最初探索之后，任何有用的测试就会被加到接受测试套件之中。

该测试有一个专业化的变种，如安全审计。在这些情况下，专业测试人员会利用他们的领域知识（可能也包括代码评审）来指导他们的测试。

机构测试

硬件产品需要不同的机构认证：FCC 度量电磁辐射，确保产品不会导致无线电干扰；美国保险商实验室（UL）检查当你将产品置于火上或舔电池电极时会发生什么。这些测试都在新产品发布之前进行，每次硬件变化都会影响认证。

环境测试

硬件产品的运行温度和湿度也需要在推至极限时测试。这些测试是用环境室来完成的，它可以同时控制这两个因素；当产品在其间运行时，它会经历所有四种极限条件。