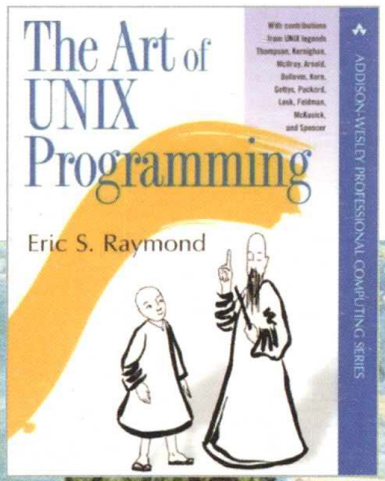
Broadview®
www.broadview.com.cn
PEARSON

传世经典书丛
Eternal Classics

UNIX编程艺术



[美] Eric S. Raymond 著
姜宏 何源 蔡晓骏 译

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

UNIX 编程艺术

The Art of UNIX Programming

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书主要介绍了 Unix 系统领域中的设计和开发哲学、思想文化体系、原则与经验，由公认的 Unix 编程大师、开源运动领袖人物之一 Eric S. Raymond 倾力多年写作而成。包括 Unix 设计者在内的多位领域专家也为本书贡献了宝贵的内容。本书内容涉及社群文化、软件开发设计与实现，覆盖面广、内容深邃，完全展现了作者极其深厚的经验积累和领域智慧。

版权贸易合同登记号：图字：01-2003-7697

图书在版编目（CIP）数据

UNIX 编程艺术 /（美）理曼德（Raymond,E.S.）著；姜宏，何源，蔡晓骏译. —北京：电子工业出版社，2011.1
（传世经典书丛）
书名原文：The Art of UNIX Programming
ISBN 978-7-121-12329-0

Ⅰ. ①U… Ⅱ. ①理… ②姜… ③何… ④蔡… Ⅲ. ①UNIX 操作系统－程序设计 Ⅳ. ①TP316.81

中国版本图书馆 CIP 数据核字(2010)第 223863 号

责任编辑：周 筠
文字编辑：许 艳
印 刷：
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：35.25 字数：650 千字
印 次：2011 年 1 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。

悦读上品 得乎益友

孔子云："取乎其上，得乎其中；取乎其中，得乎其下；取乎其下，则无所得矣"。

对于读书求知而言，这句古训教我们去读好书，最好是好书中的上品——经典书。其中，科技人员要读的技术书，因为直接关乎客观是非与生产效率，阅读选材本更应慎重。然而，随着技术图书品种的日益丰富，发现经典书越来越难，尤其对于涉世尚浅的新读者，更为不易，而他们又往往是最需要阅读、提升的重要群体。

所谓经典书，或说上品，是指选材精良、内容精练、讲述生动、外延丰盈、表现手法体贴入微的读品，它们会成为读者的知识和经验库中的重要组成部分，并且拥有从不断重读中汲取养分的空间。因此，选择阅读上品的问题便成了有效阅读的首要问题。当然，这不只是效率问题，上品促成的既是对某一种技术、思想的真正理解和掌握，同时又是一种感悟或享受，是一种愉悦。

与技术本身类似，经典 IT 技术书多来自国外。深厚的积累、良好的写作氛围，使一批大师为全球技术学习者留下了璀璨的智慧瑰宝。就在那个年代即将远去之时，无须回眸，也能感受到这一部部厚重而深邃的经典著作，在造福无数读者后从未蒙尘的熠熠光辉。而这些凝结众多当今国内技术中坚美妙记忆与绝佳体验的技术图书，虽然尚在国外图书市场上大放异彩，却已逐渐淡出国人的视线。最为遗憾的是，迟迟未有可以填补空缺的新书问世。而无可替代，不正是经典书被奉为圭臬的原因？

为了不让国内读者，尤其是即将步入技术生涯的新一代读者，就此错失这些滋养过先行者们的好书，以出版 IT 精品图书，满足技术人群需求为己任的我们，愿意承担这一使命。本次机遇惠顾了我们，让我们有机会携手权威的 Pearson 公司，精心推出"传世经典书丛"。

在我们眼中，"传世经典"的价值首先在于——既适合喜爱科技图书的读者，也符合专家们挑剔的标准。幸运的是，我们的确找到了这些堪称上品的佳作。丛书带给我们的幸运颇多，细数一下吧。

得以引荐大师著作

有恐思虑不周，我们大量参考了国外权威机构和网站的评选结果，并得到了 Pearson 的专业支持，又进

一步对符合标准之图书的国内外口碑与销售情况进行细致分析，也听取了国内技术专家的宝贵建议，才有幸选出对国内读者最富有技术养分的大师上品。

向深邃的技术内涵致敬

中外技术环境存在差异，很多享誉国外的好书未必适用于国内读者；且技术与应用瞬息万变，很容易让人心生迷惘或疲于奔命。本丛书的图书遴选，注重打好思考方法与技术理念的根基，旨在帮助读者修炼内功，提升境界，将技术真正融入个人知识体系，从而可以一通百通，从容面对随时涌现的技术变化。

翻译与评注的双项选择

引进优秀外版著作，将其翻译为中文供国内读者阅读，较为有效与常见。但另有一些外语水平较高、喜好阅读原版的读者，苦于对技术理解不足，不能充分体会原文表述的精妙，需要有人指导与点拨。而一批本土技术精英经过长期经典熏陶及实践锤炼，已足以胜任这一工作。有鉴于此，本丛书在翻译版的同时推出融合英文原著与中文点评、注释的评注版，供不同志趣的读者自由选择。

承蒙国内一流译(注)者的扶持

优秀的英文原著最终转化为真正的上品，尚需跨越翻译鸿沟，外版图书的翻译质量一直屡遭国内读者诟病。评注版的增值与含金量，同样依赖于评注者的高卓才具。好在，本丛书得到了久经考验的权威译(注)者的认可和支持，首肯我们选用其佳作，或亲自参与评注工作。正是他们的参与保证了经典的品质，既再次为我们的选材把关，更提供了一流的中文表述。

期望带给读者良好的阅读体验

一本好书带给人的愉悦不止于知识收获，良好的阅读感受同样不可缺少，且对学业不无助益。为让读者收获与上品相称的体验，我们在图书装帧设计与选材用料上同样不敢轻率，惟愿送到读者手中的除了珠玑章句，还有舒适与熨帖的视觉感受。

所有参与丛书出版的人员，尽管能力有限，却无不心怀严谨之心与完美愿望。如果读者朋友能从潜心阅读这些上品中偶有获益，不啻为对我们工作的最佳褒奖。若有阅读感悟，敬请拨冗告知，以鼓励我们继续在这一道路上贡献绵薄之力。如有不周之处，也请不吝指教。

电子工业出版社博文视点

二〇一〇年十二月

给 Ken Thompson 和 Dennis Ritchie，

你们激励了我。

译序

大多数译序是给作者说好话，顺便带动一下译本销量的，本篇是一个例外。

《The Art of UNIX Programming》，简称 TAOUP，作者 Eric S. Raymond，简称 ESR。这大概是计算机类书籍中很少见的一本课外读物。TCP/IP 编程之类典型 Unix 编程书中讲到的东西在这本书里面找不到，所以书里讲到的当然就是别的书里找不到的东西。读者也许需要有相当的 Unix 背景、或者长期钻研某个专题，才能体会到作者的弦外之音。ESR 作为老牌黑客信手拈来的典故，如果不是在 Unix 里面长期浸淫，大概很难有所共鸣，所以把这当作 Unix 的一部坊间史话倒也合适。

本书总结了历史上 Unix 众多成功的经验和失败的教训、经时间考验和临时搭救的编码策略、大众喜爱和小众受用的实用工具；一些被跨国界信仰地广泛接受，一些则在不同环境中各有见地。被 TAOUP 总结为失败的，也许恰恰是某些工程的保命神药；总结为成功的，也许正好是压垮另一些工程的最后一根稻草。情景各异而已。书是写给程序员看的，因此很多观点都太过技术味儿，比如所见即所得的编辑器不如手写标记的纯文本更直接——90%的人会想：这怎么可能？！

这本书是给读者增长见识的，很多案例分析不管结论如何，读者都可以从中见到红蓝两方的思维方式和行事方法，以及各方高手看待问题的角度。无论成功还是失败，都只是一念之间，而读者只需要体味出这些对自己过去的、手头的、未来的项目可以有何种借鉴，便已得其中三昧。

网络上关于 TAOUP 的书评甚多，正负反响各有不少，负面评价大体集中在认为作者视角较窄、对商业公司有偏见以及过分抬爱自己的 fetchmail 几方面。我个人的感觉，Unix、尤其是开源 Unix 上有太多好用的工具极欠雕琢，目标受众太过技术。ESR 并未回避这些，读者不妨多留意为数不多的痛切之笔。

本书翻译经历一年多的时间，之前我曾经约略翻过纸版，偶尔见到一些合我胃口的言论，于是心有灵犀认为这书不错；然而等到译到中途，便发现 ESR 实在是个美国愤青，这便是课外读物和工本教程给读者的不同感受了。翻译的过程对译者是精读的过程，但希望读者能用它打发堵车、候机、等人时的无聊时间，这书适合从任何一篇翻起。

翻译过程颇为艰辛：何蔡两位初译，由我统稿。书中寻章摘句之处，我们尽力将其还原。书名保持原文并给出译名，人名不译，专有名词给出原文，特意不加入任何译注。相关背景常识、翻译感受以及付梓后的任何问题，可以在中译版网页上与我们交流。这一年间，侯捷老师的推荐，周筠老师、方舟和兴璐两位编辑、何蔡二位给我的莫大帮助和宽容使得本书最终面世；身边诸位好友同事也不同程度地在各个技术方面给予指导和支持，尤其感谢 bz、主任、delphij、kola 几位。我的爱人王冰陪我加班，容忍我对程序的沉迷，给我心灵的温暖，是我翻译这本书的力量源泉。

KISS。

姜 宏

2005 年 12 月于北京

序

Preface

Unix is not so much an operating system as an oral history.
与其说 Unix 是个操作系统，不如说是一部口述历史。
——Neal Stephenson

知识和专能差异巨大，凭借知识可以推断出该做什么，而专能让你甚至在无意之间，条件反射似的把事情做好。

这本书确实有关“知识”，但更着眼于“专能”。你将学到那些 Unix 专家们都不自知的 Unix 开发知识。少一点技术，多一些共享文化：显见和隐微的，直观和潜流的——这是本书和大多数 Unix 书籍不同的地方——不止于方法，更重乎理念。

理念于实用大有裨益，有太多设计不良的软件：体积臃肿，难于维护、移植和扩展——这些都是蹩脚设计的症候。我们希望本书的读者能品出什么是 Unix 所教示的良好设计。

本书分为四部分：场景、设计、工具和社群。第一部分（场景）涉及哲学和历史，为后续内容埋下伏笔。第二部分（设计）将 Unix 哲学的原理细分为有关设计与实现的、更专门的建议。第三部分（工具）着眼于 Unix 所提供的工具，可助你解决问题。第四部分（社群）则讲述人与人之间的事务与约定，而这正是 Unix 文化拥有高效能的原因。

这本书是关于共享文化的，我从未想像过独自完成它。你会发现正剧中包含数位 Unix 资深专家的客串演出，正是这些人塑造了 Unix 的习俗。本书曾有过公开大范围的审阅过

程，这期间我邀请这些明星人士对书稿进行评审与研论。这些意见没有湮没在本书定稿中，而你可以在书中聆听到他们的真实声音：无论是为本书呐喊助阵、还是摇头反对。

本书中用到人称“我们”时，我并不是虚张声势，仅以此说明这是整个社群都清楚明了的事实。

因为本书着力传递文化，因此加入了很多野史和坊间传说，这在技术书中并不多见。希望你喜欢，这些东西其实是 Unix 程序员的教养。须弥不重，芥子不轻。我们希望以这种方式更好地讲述故事。了解 Unix 的由来和变迁，会培养你对 Unix 风格的直觉。

同样地，基于此，我们不打算使用回述历史的腔调。你会发现本书参考了众多时下信息。我们不希望给你一种错觉：书里说的都是亘古不变的终极真理。参考时下的信息这一做法，也提醒读者，三十年河东，三十年河西，眼前所见，也许过不了多久就会过时，而需要重新检省。

另外，本书不是 C 教程，不是 Unix 命令和 API 的手册，不是 sed/yacc/Perl/Python 的语言参考，也不是网络编程入门，更不是巨细靡遗的令人费解的 X 指南。本书也不打算带你巡游 Unix 内幕和体系。有很多其它的好书涵盖这些领域，本书会在适当的时候告诉你该看哪些。

在这些技术细节外，Unix 文化有一个未见诸笔端的行工传统，以熟练工的考量，它已经有几百万人年的发展[1]。本书即立足于这样一个信念：领会此传统，并将它的设计手法应用到手边，你将成为更好的程序员和设计师。

构成文化的是人，一直以来，获知文化的方式大约是口口相传、潜移默化。本书不打算取代人际的文化传播，但可以促进这一过程，使你能俯耳倾听他人的心声。

[1] 从 1969 到 2003 年，35 年时间可不算短。以这期间 Unix 站点数量的年度曲线计算，人们在 Unix 上耕作了约有 5000 万人年。

谁应该看这本书

如果你是个 Unix 编程老手，经常教导菜鸟，或者与人进行操作系统论战时无法阐明使用 Unix 方案所带来的好处时，可以看看这本书。

如果你是个 C、C++或者 Java 程序员，有其它操作系统的开发经验，现在轮到你开展一个 Unix 项目时，可以看看这本书。

如果你是个初级或者中级水平的 Unix 用户，但是没什么开发经验，想学习在 Unix 下如何高效地设计软件时，可以看看这本书。

如果你不在 Unix 下编程却发觉 Unix 的传统给你带来某种启迪，那你就对了，Unix 哲学适用于其它的操作系统。因此我们会花比其它 Unix 书籍更多的篇幅关注非 Unix 环境（特别是微软的操作系统）；当所用到工具或者案例可用于其它操作系统时，我们会告诉你。

如果你是一个系统架构师，正为通用市场或垂直应用准备平台方案或实现策略时可以看看这本书。本书将帮助你了解 Unix 作为开发平台的强大功能，以及开放源码这个 Unix 的传统所带来的开发方式。

如果你想学到 C 编程的细节或者想知道怎么用 Unix 内核 API，本书可能不适合你。Advanced Programming in theUnix Environment [Stevens92]是探究 Unix API 的经典名著；The Practice of Programming [Kernighan-Pike99]是每个 C 程序员的必读书目（任何语言的程序员都该看看这本书）。

如何使用这本书

这本书既重实践，更富理念；既包含警世格言，又不忘检点 Unix 开发中的特殊案例。在每个警句前后，都有生动实例阐明其由来，这些例子绝不来自小儿科式的示例程序，而均出自真实世界满眼所见的运行代码。

我们着力避免以大量代码或者规范文件来胡乱凑数，当然这么做会让本书的写作轻松许多（某些地方或许读起来也更轻松）。绝大多数编程书籍只授你以鱼，而本书避免这种做法，力求培养读者“探求事情何以如此”的感知力。

正由于此，本书会时常请你阅读代码与规范文档，它们中极少量的内容会附在书中，其余部分我们会在举例时告诉你如何从网上获取。

从这些范例中汲取养分，将有助你将所学原则消化变为庖丁之技。如果你能就着一部跑在 Unix 系统上的网页浏览器来读书，是再理想不过的了。任何 Unix 系统都适合，但是我们将要研究的案例大多都会预装在、或者可以从 Linux 系统上获得，书中会提示请你浏览或亲身感受它们。这些提示通常是按部就班的，跑开玩一会儿并不会打散整个讲述过程的连续性。

注意：我们虽力求，但无法给你打保票，声称我们所引用的 URLs 稳定可用。如果你发现某个引用连接已陈旧过时，来点常识，用你喜爱的搜索引擎来个短语搜索。如有可能，我们会在所引用的 URLs 附近给出如何搜索的提示。

大多数缩写形式会在首次出现时伴随其全称。为方便起见，我们在附录中提供了名词对照表。

交叉索引通常以作者名字为主导词。带编号的脚注是那些可能会扰乱你阅读正文，或者是易变的 URLs；也可能是旁征博引的战争故事或者笑话[2]。

为了使这本书不至于让非技术人员太过难读，我们邀请了一些非程序员试读，并指出一些晦涩但起贯穿作用的词汇。我们把那些编程老手不太会需要的名词解释也放在脚注中。

相关引文

一些 Unix 早期拓荒者的著名论文和书籍，比如 Kernighan 和 Pike 的《The Unix Programming Environment》[Kernighan-Pike84]就是其中佼佼者，被世人尊为圭臬。而今看来此书廉颇老矣，它没提到 Internet、万维网以及诸如 Perl、Tcl 和 Python 这些解释型语言的新秀。

写作本书的中途我们借鉴了 Mike Gancarz 的《The Unix Philosophy》[Gancarz]。这本书在它的覆盖范围内极其优秀，但是我们觉得需要更多内容才能反映出事情的全貌。尽管如此我们仍对此书作者心存感激，他愈发使我们知道最简单的 Unix 设计手法就是最持久耐用的。

[2] 这个特别的脚注献给 Terry Pratchett，他对脚注的用法简直是……绝了。

《The Pragmatic Programmer》[Hunt-Thomas]是一本关于良好设计的书，文风机智诙谐，它与本书相比，倾向于软件设计工艺的另一个层面（更注重编码，而少着墨于高层面的问题划分）。作者的哲学是其 Unix 领域耕耘的成果，也是本书内容极好的补充。

《The Practice of Programming》[Kernighan-Pike99]包含了一些与《The Pragmatic Programmer》共通的内容，但更钻入 Unix 传统的深处。

最后（明知道会激怒你），我们推荐《Zen Flesh, Zen Bones》[Reps-Senzaki]，一部重要的佛教禅宗本源的合集。对禅的引用书目遍布全书。我们将这些书目包含进来，是因为禅为表达某种想法提供了丰富的语汇，而在软件设计中却很难烂熟于心。信奉宗教的读者，请您不要把禅当成宗教，它是一种心灵鸡汤似的东西，纯净而没有神灵的干扰——此即是禅。

本书的习俗约定

术语“UNIX”技术上和法律上讲，是 The Open Group 的商标，并且应该仅限于那些通过 The Open Group 严格的“符合标准”认证的操作系统。本书中我们使用其较宽松的定义，即大多数程序员所指的，Bell 实验室 Unix 代码的后裔或旁支。在这个意义下，Linux（大多数例子都举自它）也算是一种 Unix。

本书也使用了 Unix 手册页（manual page）的传统，即以括号括起来的手册节号来标记 Unix 设施。通常用于强调一个 Unix 命令首次出现。比如“munger(1)”可解读为“munger 程序加入存在于你的系统中，其文档位于 Unix 手册页的第 1 节”。第 2 节是 C 的系统调用，第 3 节是 C 的库函数调用，第 5 节是文件格式与协议，第 8 节是系统管理工具。其它节号本书未曾用到，其定义在各个 Unix 系统各有不同。在你的 Unix 外壳提示符下输入 `man 1 man`（老式的 System VUnix 系统可能要输入 `man -s 1 man`）以获得更多信息。

有时我们会提及某个 Unix 程序（比如 *Emacs*），后面没有手册节号而且首字母大写。这意味这个名字代表一族 Unix 程序，其基本功能相同，而我们将讨论其通用特性。比如 *Emacs*，就包含了 *xemacs*。

本书很多地方我们同时给出了老式(old school)和新式(new school)解法。new-school 和 rap 音乐一样，开始于 1990 年前后。在这个含义下，我们往往把它与脚本语言、图形

用户界面、开放源码的 Unix 和万维网联系起来。Old-school 指代 1990 年以前（特别是 1985 年以前）的世界：昂贵的共用计算机、专属的 Unix，shell 脚本和无所不在的 C。值得指出这些差异，机器越来越便宜，内存多了起来，这些有如暗流，渐渐影响着 Unix 编程的风格。

所用案例

很多编程书籍为证明某一观点而特地造出一个范例，你手中这本书不这么干。我们的案例研究均来自真实世界，在生产环境中工作已久。下面是一些主要案例：

cdrtools/xcdroast

这两个独立的项目通常被一并使用。cdrtools 是一组刻盘工具（用关键字“cdrtools”可以在网上找到）。xcdroast 是 cdrtools 的图型界面前端，其项目网站为 <http://www.xcdroast.org/>。

fetchmail

fetchmail 用于从远程邮件服务器上收信，支持 POP3 和 IMAP 邮箱协议。其主页为 <http://www.catb.org/~esr/fetchmail>，也可以用关键字“fetchmail”从网上找到。

GIMP

GIMP（GNU Image Manipulation Program，GNU 图像处理程序）是一个全特性的绘画和图像处理程序，可对多种图像格式进行复杂处理。其源码可从 GIMP 主页 <http://www.gimp.org/>获得（也可以通过关键字“GIMP”从网上搜到）。

mutt

mutt 邮件客户端是目前各类基于文本的邮件客户端程序中的翘楚，提供对 MIME（Multipurpose Internet Mail Extensions）、个人隐私辅助程序，如 PGP（Pretty Good Privacy）和 GPG（GNU Privacy Guard）等特性的绝佳支持。其源码和二进制可执行文件可以从 Mutt 项目主页<http://www.mutt.org>获得。

xmlto

xmlto 可将 DocBook 和其它 XML 文档以多种格式渲染输出，包括 HTML、纯文本

和 PostScript。其源码和文档可在 xmlto 主页<http://cyberelk.net/tim/xmlto/>获得。

为了将读者理解本书例子所要阅读的代码量降低到最小程度，我们尽量挑选那些可重复使用、并能体现多种不同设计原理和设计实践的案例。出于同样原因，很多示例来自于我本人的项目。我没想说这些例子最为恰当，只是我觉得它们对阐述我的观点非常有用。

作者致谢

各位客串贡献者（Ken Arnold, Steven M. Bellovin, Stuart Feldman, Jim Gettys, Steve Johnson, Brian Kernighan, David Korn, Mike Lesk, Doug McIlroy, Marshall Kirk McKusick, Keith Packard, Henry Spencer, 和 Ken Thompson）为本书增添极大价值。特别是 Doug McIlroy，给予本书恪尽职责、鞭辟入里的评注的同时，也展现了他早在 30 年前管理最原始的 Unix 研究组时鞠躬尽瘁的高风亮节。

我要对 Rob Landley 和我的妻子 Catherine Raymond 致以特别感谢，他们都不厌其烦地逐行对本书手稿进行审阅。Rob 的深富洞察力的细致评述激励我在最终稿中加入了一整章内容，他为本书的组织结构与取材范围奉献极多。如果把他所给予的改进意见落在笔端，那他无愧于本书的合著者。Cathy 代表读者中非技术人员的一群，如果那些非程序员读者觉得本书并不难读，那全是她的功劳。

写作的五年间，本书从不少人的讨论意见中获益良多。Mark M. Miller 使我对线程有了更深的认识。John Cowan 教给我不少接口设计方式，并起草了 wily 和 VM/CMS 的学习案例。Jef Raskin 告诉我 Rule of Least Surprise 的由来。UIUC System Architecture Group 对前几章给出的反馈弥足珍贵，What Unix Gets Wrong 和 Flexibility in Depth 两节是他们直接激励的结果。Russell J. Nelson 提供了 Bernstein chaining 的素材。第 3 章中 MVS 学习案例大部分的材料来自 Jay Maynard。Les Hatton 对语言一章给出很多有益建议，并促使我写成第 4 章中 Optimal Module Size 的部分内容。David A. Wheeler 贡献了很多发人深省的批评，以及一些学习案例（特别是在设计部分中）的素材。Russ Cox 帮助我进行了 Plan 9 的调查。Dennis Ritchie 纠正了我的一些错误的 C 历史观念。

成百上千的 Unix 程序员，人数太多以至于无法在此列出他们的名字，在 2003 年 1 月到 6 月间的公开审阅过程间给了我建议和评论。开放的同级复审这一过程让我觉得紧张刺激而回报极多。当然，任何最终书稿中残留的错误都是我自己的责任。

“把事情说透”的风格，以及其它一些考虑因素，是受到了“设计模式运动”的影响；说实话，我对到处堆砌 Unix 设计模式这种做法深不以为然。我对此运动的中心教条不敢苟同，并且没觉得把设计模式严格付诸实用有什么必要，也不想背上这种思想的包袱。尽管如此，我的行事方法仍然受到 Christopher Alexander 成果[3]（特别是《Timeless Way of Building》和《A Pattern Language》两文）的影响。Gang of Four 和他们的信徒为我展示了如何用 Alexander 的思想，站在较高层面上，抛去含混不清的对设计通则的空话，来谈论软件设计，这一点我心存感激，永志不忘。对设计模式有兴趣的读者可以看看这本书《Design Patterns: Elements of Reusable Object-Oriented Software [GangOfFour]》。

本书标题毫无疑问是借鉴了 Donald Knuth 的《The Art of Computer Programming》一书的书名。Knuth 和 Unix 传统文化没什么联系，但他影响了我们每一个人。

有先见之明和丰富想象力的编辑并不多，好在 Mark Taub 就是一个，他从并不看好的项目中发现了闪光点，并极富技巧地促成了这本书的写作。文字编辑中，文笔好而又能帮助别人提高文笔的就更少了，所幸 Mary Lou Nohr 是其中之一。Jerry Votta 的封面设计领会了我的意图，而且做得比我的想像还要漂亮。Addison-Wesley 的编辑们让审稿和出版这一过程不再枯燥无味，我天生怕被人管，但是他们仍然极力配合我，使得文字、版面、图片和市场工作都达到极高水准。

[3] 一篇对 Alexander 工作成果的肯定文章，包含其部分重要成果的在线连接，可以参考 Some Notes on Christopher Alexander <http://www.math.utsa.edu/sphere/salingar/Chris.text.html>一文。

Contents