21st Century C

C程序设计新思维

[美] *Ben Klemens* 著
赵铁成 徐波 译

O'REILLY®
人民邮电出版社
POSTS & TELECOM PRESS

O'REILLY®

C 程序设计新思维

[美] Ben Klemens 著
赵铁成 徐 波 译

人 民 邮 电 出 版 社

图书在版编目（CIP）数据

C程序设计新思维 / （美）克莱蒙（Klemens, B.）著 ；赵铁成，徐波译. — 北京 ： 人民邮电出版社，2015.5
ISBN 978-7-115-38628-1

Ⅰ. ①C… Ⅱ. ①克… ②赵… ③徐… Ⅲ. ①C语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2015)第060205号

版权声明

◆ 著　[美] Ben Klemens
译　赵铁成　徐　波
责任编辑　陈冀康
责任印制　张佳莹　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
三河市海波印务有限公司印刷

◆ 开本：787×1000　1/16
印张：18
字数：341 千字　2015 年 5 月第 1 版
印数：1 – 3 000 册　2015 年 5 月河北第 1 次印刷
著作权合同登记号　图字：01-2013-3681 号

定价：49.00 元

读者服务热线：(010) 81055410　印装质量热线：(010) 81055316
反盗版热线：(010) 81055315

内容提要

C 语言已经有 40 年的历史了。经过长时间的发展和普及，C 语言的应用场景有了很大的变化，一些旧观念应该被淡化或者不再被推荐。

本书展现了传统 C 语言教科书所不具有的最新的相关技术。全书分为开发环境和语言两个部分，从编译、调试、测试、打包、版本控制等角度，以及指针、语法、文本、结构、面向对象编程、库等方面，对 C 程序设计的核心知识进行查缺补漏和反思。本书鼓励读者放弃那些对大型机才有意义的旧习惯，拿起新的工具来使用这门与时俱进的简洁语言。

本书适合有一定基础的 C 程序员或 C 语言学习者阅读，也适合想要深入理解 C 语言特性的读者参考。

前言

Is it really punk rock

Like the party line?

它真的是朋克摇滚么，

就像政治路线？

——选自 Wilo 的歌曲 “Too Far Apart（遥远）”

C 就是 Punk Rock

C 仅有为数不多的关键词，并且显得略微粗糙，但是它很棒[1]！你可以用 C 来做任何事情。它就像一把吉他上的 C、G 和 D 弦，你可以很快就掌握其基本原理，然后就得用你的余生来打磨和提高。不理解它的人害怕它的威力，并认为它粗糙得不够安全。实际上它在所有的编程语言排名中一直都被认为是最流行的语言，所以根本不需要任何企业或组织花钱去推广它。

这门语言已经有 40 年的历史了，可以说已经进入了中年。创造它的人是少数对抗管制的人，遵从完美的 punk rock 精神；但那是 20 世纪 70 年代的事情了，现在它已历尽沧桑，成为社会的主流。

当 punk rock 变成主流的时候人们会怎样？在其从 20 世纪 70 年代出现后的几十年里，punk rock 已经从边缘走向中心：The Clash、The Offspring、Green Day 和 The Strokes 等乐队已经在全世界卖出了几百万的唱片（此处仅作有限的举例），我也在家附近的超市里听过被称为 grunge 的一些精简乐器版本的 punk rock 分支。Sleater-Kinney 乐队的前主唱还经常在自己那个很受欢迎的喜剧节目中讽刺 punk rocker 音乐人。对这种持续的进化，一种反应是画一条界限，将原来的风格称为 punk

1 it rocks，此处原文为双关语，借用英语中 rock 的不同含义，即“摇滚乐”和“很棒”。标题中的 punk rock 为流行于 20 世纪 70 年代的一种摇滚乐风格，以狂野反叛为特色，国内也称为“朋克”。——译者注

rock，而将其余的东西称为面向大众的粗浅的 punk。传统主义者还是可以播放来自 20 世纪 70 年代的唱片，但如果唱片的音轨磨损了，他们可以购买数码修复的版本，就像他们为自己的小孩购买 Ramones 牌的连帽衫一样。

外行是不明白的。有些人听到 punk 这个词脑海里就勾画出 20 世纪 70 年代并不具有的景象，经常的历史错觉就是那个时代的孩子们真的在做什么不同的事情。喜欢欣赏 1973 年 Iggy Pop 的黑胶唱片的传统主义者一直是那么兴趣盎然，但是他们有意无意地加强了那种 punk rock 已经停滞不前的刻板印象。

回到 C 的世界里，这里既有挥舞着 ANSI'89 标准大旗的传统主义者，也有那些拥抱变化，甚至都没有意识到如果回到 20 世纪 90 年代，他们写的代码都不可能被成功编译与运行的人。外行人不会知道个中缘由。他们看到从 20 世纪 80 年代起至今还在印刷的书籍和 20 世纪 90 年代起至今还存于网上的教程，他们听到的都是坚持当年的软件编写方式的、死硬的传统主义者的言论，他们甚至都不知道语言本身和别的用户都在一直进化。非常可惜，他们错过了一些好东西。

这是一本打破传统并保持 C 语言 punk 精神的书。我对将本书的代码和 1978 年 Kernighan 和 Ritchie 出版的书[1]中的 C 标准进行对比毫无兴趣。既然连我的电话机都有 512M 字节内存，为什么还在我的书里花费章节讲述如何为可执行文件减少几 K 的字节呢？我正在一个廉价的红色上网本上写这本书，而它却可以每秒运行 3 200 000 000 条指令，那为什么我还要操心 8 位和 16 位所带来的一个操作的差异呢？我们更应该关注如何做到快速编写代码并且让我们的合作者们更容易看懂。毕竟我们是在使用 C 语言，所以我们那些易读但是并没有被完美优化的代码运行起来还是会比很多繁琐的语言明显地快。

Q&A（本书的参考引用）[2]

Q：这本书与其他书有什么不同？

A：C 语言的教科书都非常相像（我曾经读过很多这样的教科书，包括[Griffiths，2012]、[Kernighan，1978]、[Kernighan，1988]、[Kochan，2004]、[Oualline，1997]、[Perry，1994]、[Prata，2004]和[Ullman，2004]）。多数教材都是在 C99 标准发布并简化了很多用法之后写成的，你可以看到现在出版的这些教材的第 N 版仅仅在一些标注上做了一点说明，而不是认真反思了如何使用这门语言。他们都提到你可以拥

1 从下文可以看到，该书的出版被认为是 C 语言诞生的标志性事件，业内常成为 K&R——译者注
2 这里作者将问题与解答比喻为 C 语言函数入口的参考引用——译者注

有一些库来编写你自己的代码，但是他们都是在现在常用的、可以保障库的可靠性和可移植性的安装与开发环境之前编写的。那些教科书现在仍然有效并且具有自己的价值，但是现代的C代码已经看起来和那些教科书里面的不太一样了。

这本书与那些教科书的不同，在于对这门语言及其开发环境进行了拾遗补漏。书中讲解的方式是，使用提供了链表结构和XML解析器的现成的库，而不是把这些从头再写一次。这本书也体现了如何编写易读代码和用户友好的函数接口。

Q：这本书的目标读者是谁？我需要是一个编程大师么？

A：你必须有某种语言的编程经验，或许是Java，或者类似Perl的某种脚本语言。这样我就没有必要再和你讲为什么你不应该写一个很长的没有任何子函数的函数了。

你最好有一定的C语言基础，但是没有必要特别精通——既然我将描述那些细节，如果你从来没有学过它们也许更好。如果你是白纸一张，只是对C语法充满敬意，那它还是非常简单易学的，而且你也可以用搜索引擎找到很多在线教材；如果你有其他语言的经验，用一两个小时就可以基本掌握。

请允许我介绍我写的另一本关于统计和科学计算的教科书，*Modeling with Data* [Klemens，2008]。那本书不仅提供了很多关于如何处理数值和统计模型的内容，它还可以用作一本独立的C语言的教材，并且我认为那本书还避免了很多早期教材的缺点。

Q：我是个编写应用软件的程序员，不是一个研究操作系统内核的人。为什么我应该用C而不是像Python这类可以快速编程的脚本语言呢？

A：如果你是一个应用软件程序员的话，这本书就是为你准备的。我知道人们经常认定C是一种系统语言，这让我觉得真是缺少了点punk的反叛精神——他是谁啊？要他告诉我们要用什么语言？

像“我们的语言几乎和C一样快，但更容易编写”这样的言论很多，简直成了刻板的套话。好吧，C显然是和C一样快，并且这本书的目的是告诉你C也比以前的教科书所暗示的那样容易使用。你没必要使用malloc，也没必要向20世纪90年代的系统程序员那样深深卷入内存管理，我们已经有处理字符串的手段，甚至核心语法也进化到了支持更易读的代码的境界。

我当初正式学习C语言是为了加速一个用脚本语言R编写的仿真程序。和众多的

脚本语言一样，R 具有 C 接口并且鼓励用户在宿主语言[1]太慢的时候使用。最终我的程序里有太多的从 R 语言到 C 语言的调用，最后我索性放弃了宿主语言。随后发生的事情你已经知道，就是我在写这本关于现代 C 语言技术的书。

Q：如果原本使用脚本语言的应用软件程序员能喜欢这本书当然好，但我是一名内核黑客。我五年级的时候就自学了 C 语言，有时做梦都在正确编译。那这本书还有什么新鲜的么？

A：C 语言在过去的 20 年里真的进化了很多。就像我下面要讨论的那样，像“要支持所有 C 编译器”等必要的工作准则也变化了不少，因为自从 ANSI 标准发布后又发布了两个新的 C 语言标准。也许你应该读一下第 10 章，找找有什么能叫你感到惊讶的。

并且，开发环境也进化升级了。Autotools 已经改变了代码发布的方式，意味着可以更加可靠地调用其他的库，意味着我们的代码可以花费更少的时间在重建常用结构和函数上，而是更多地调用本书下面将讨论的库。

Q：我实在忍不住要问，为什么这本书中差不多有三分之一的篇幅都没有 C 代码？

A：的确。良好的 C 实践需要具备精良的 C 工具[2]。如果你没有使用调试器（独立的或者集成在你的 IDE 中），你就是在自讨苦吃。如果你告诉我追踪内存的泄露是不可能的，那么就意味着你还没有听说过 Valgrind，一个用来确切地指出是哪一行产生内存泄漏并发生错误的系统。Python 及其附带工具有内建的包管理器；而本书将告诉你，属于 C 的、事实上的跨平台打包系统，即 Autotools，它是一个独立的系统。

如果你在使用一个不错的集成开发环境（IDE）作为这些工具的调用界面，你仍然可以从了解“IDE 如何处理环境变量以及其他隐藏的细节，同时还能为你处理错误抛出”等问题中受益。

Q：你所谈论的一些工具有点老了。难道没有更现代的工具能替代这些基于 shell 的工具么？

A：如果我们嘲笑那些仅仅因为事物是新的就对其抵制的人，那么我们也没有理由仅仅因为事物是旧的就加以抵制。

其实很容易找到可靠的来源证明第一件六弦的吉他出现在 1200 年左右，第一个四

1 这里指调用 C 语言的语言——译者注
2 即工欲善其事，必先利其器——译者注

弦的小提琴出现在大约 1550 年，带键盘的钢琴出现在 1700 年左右。有趣的是，你今天听到的多数（如果不是全部）音乐都与以上乐器中的某种有关。Punk rock 当初并没有拒绝吉他，只不过用得更加有创造性，比如将吉他的输出接连到新的滤波器上。

Q：我能上 Internet，一两秒的功夫就能找到命令和语法的细节。那么说真的，为什么我还要读这本书？

A：的确。在 Linux 或 Mac 机器上你只要用一个带有 man operator 的命令行就能查到运算符优先级表，那么我为什么还要把它放在这本书里？

我可以和你上同样的 Internet，我甚至花了很多的时间阅读网上的内容。所以我有了一个之前没谈到的、准备现在讲的好主意：当介绍一个新工具的时候，比如 gprof 或者 GDB，我给你那些你必须知道的方向，然后你可以去自己习惯的搜索引擎中查找相关问题。这也是其他教科书所没有的。

标准：难以抉择

除非特别地说明，本书的内容遵从 ISO C99 和 C11 标准。为了使你明白这意味着什么，下面给你一点 C 语言的历史背景，让我们回顾一下主要的 C 标准（而忽略一些小的改版和订正）。

K&R（1978 前后）

Dennis Ritchie、Ken Thompson 以及一些其他的贡献者发明了 C 语言并编写了 Unix 操作系统。Brian Kernighan 和 Dennis Ritchie 最终在他们的书中写下了第一版关于这个语言的描述，同时这也是 C 语言的第一个事实上的标准[Kernighan，1978]。

ANSI C89

后来 Bell 实验室向美国国家标准协会（ANSI）交出了这个语言的管理权。1989 年，ANSI 出版了他们的标准，并在 K&R 的基础上做出了一定的提高。K&R 的书籍的第二版包含了这个语言的完整规格，也就是说在几万名程序员的桌子上都有这个标准的印刷版[Kernighan，1988]。1990 年 ANSI 标准被 ISO 基本接受，没有做重大的改变，但是人们似乎更喜欢用 ANSI 89 这个词来称呼这个标准（或者用来做很棒的 T 恤衫标语）。

10 年过去了。C 成为了主流，考虑到几乎所有的 PC、每台 Internet 服务器的基础代码或多或少都是用 C 编写的，这已经是人类的努力可以达到的最大的主流了。

在此期间，C++分离出来并大获成功（虽然也不是那么大）。C++是 C 身上发生的最好的事情了。当所有其他的语言都在试图添加一些额外的语法以跟随面向对象的潮流，或者跟随其作者脑袋里的什么新花招的时候，C 就是恪守标准。需要稳定和可移植性的人使用 C，需要越来越多特性以便可以像在百元大钞里洗澡一样无休止地沉溺于其中的人使用 C++，这样每个人都很高兴。

ISO C99

10 年之后 C 标准经历了一次主要的改版。为数值和科学计算增添了一些附加功能，比如复数的标准数据类型以及相关的函数。一些从 C++中产生的便利措施被采纳，包括单行注释（实际上起源于 C 的前期语言，BCPL），以及可以在 for 循环的开头声明变量。因为一些新添加的关于如何声明和初始化的语法，以及一些表示法上的便利，使用泛型函数变得更加容易。出于安全考量以及并不是所有人都说英语原因，一些特性也被做了调整。

当你想着单单 C89 的影响就有那么大，以及全球是如何运行 C 代码时，你很难想出 ISO 能够拒绝某种新事物而不被广泛批评——甚至拒绝做出改变也会被唾骂。的确，这个标准是有争论的。有两种常用的方式来表达一个复数（直角坐标和极坐标）——那么 ISO 会采用哪一个？既然所有的好代码都没采用变长的宏输入机制来编写，为什么我们还需要这个机制？换句话说，纯洁主义者批评 ISO 是在背叛那些为 C 语言增加更多特性的趋势。

当我写这本书的时候，多数的编译器在支持 C99 的同时都增加或减少了一些特性；比如 long double 类型看起来就引发了很多问题。然而，这里还是有一个明显的特例：Microsoft 至今拒绝在其 Visual Studio C++编译器中添加 C99 支持。在“1.2 在 Windows 中编译 C”节中讲述了几种在 Windows 环境中编译 C 的方法，所以不能使用 Visual Studio 最多也就是有点不方便，并且这好比一个行业奠基人告诉我们不能使用 ISO 标准的 C，而只能支持 punk rock。

C11

觉察到了对所谓背叛行业趋势的批评，ISO 组织在第三版的标准中做出了为数不多的几个重大改变。我们有了可以编写泛型函数的方法，并且对安全性和非英语支持做出了进一步的改进。

我是在 2012 年写的这本书，就在 C11 标准在 2011 年 12 月发布之后的不久，此时已经有了一些来自编译器和库的支持。

POSIX 标准

事物的规律就是这样，伴随着 C 语言的进化，这门语言同时也和 Unix 操作系统一起协同发展，并且你将会从本书中看到，这种相互协同的发展对日常工作是有意义的。如果某件事情在 Unix 命令行中很容易利用，那么很有可能是因为这件事情在 C 中也很容易实现；某些 Unix 工具之所以存在也是为了帮助 C 代码的编写。

Unix

C 和 Unix 都是在 20 世纪 70 年代由 Bell 实验室设计的。在 20 世纪的多数时间里，Bell 一直面临垄断调查，并且 Bell 有一项与美国联邦政府达成的协议，就是 Bell 将不会把自身的研究扩张到软件领域。所以 Unix 被免费发放给学者们去剖析和重建。Unix 这个名字是一个商标，原本由 Bell 实验室持有，但随后就像一张棒球卡一样在数家公司之间转卖。

随着其代码被不断检视、重新实现，并被黑客们以不同的方式改进，Unix 的变体迅速增加。因此带来了一点不兼容的问题，即程序或脚本变得不可移植，于是标准化工作的迫切性很快就变得显而易见。

POSIX

这个标准最早由电气电子工程师协会（IEEE）在 1988 年建立，提供了一个类 Unix 操作系统的公共基础。它定义的规格中包括 shell 脚本如何工作，像 ls、grep 之类的命令行应该如何工作，以及 C 程序员希望能用到的一些 C 库等。举个例子，命令行用户用来串行运行命令的管道机制被详细地定义了，这意味着 C 语言的 popen（打开管道）函数是 POSIX 标准，而不是 ISO C 标准。POSIX 标准已经被改版很多次了；本书编写的时候是 POSIX：2008 标准，这也是当我谈到 POSIX 标准的时候所指代的。POSIX 标准的操作系统必须通过提供 C99 命令来提供 C 编译器。

这本书用到 POSIX 标准的时候，我会告诉大家。

除了来自 Microsoft 的一系列操作系统产品，当前几乎所有你可以列举出的操作系统都是建立在 POSIX 兼容的基础上：Linux、Mac OS X、iOS、WebOS、Solaris、BSD——甚至 Windows Servers 也提供 POSIX 子系统。对于那些例外的操作系统，“1.2 在 Windows 中编译 C”节将告诉你如何安装 POSIX 子系统。

最后，有两个 POSIX 的实现版本因为有较高的流行度和影响力，值得我们注意。

BSD

在 Bell 实验室发布 Unix 给学者们剖析之后，加州大学伯克利分校的一群好人做了很多明显的改进，最终重写了整个 Unix 基础代码，产生了伯克利软件发行版（Berkeley Software Distribution，BSD）。如果你正在使用一台 Apple 公司生产的电脑，你实际上在使用一个带有迷人图形界面的 BSD。BSD 在几个方面超越了 POSIX，因此我们还会看到，有一两个函数虽然不属于 POSIX，但是如此有用而不容忽略（其中最重要的救命级函数是 asprintf）。

GNU

这个缩写代表 GNU's Not Unix，代表了另一个独立实现和改进 Unix 环境的成功故事。大多数的 Linux 发行版使用 GNU 工具。有趣的是，你可以在你的 POSIX 机器上使用 GNU 编译器组合（GNU Compiler Collection，gcc）——甚至 BSD 也用它。并且，gcc 对 C 和 POSIX 的几个方面做了一点扩充并成为事实上的标准，当本书中需要使用这些扩充的时候我会加以说明。

从法律意义上说，BSD 授权比 GNU 授权稍微宽容。由于很多群体对这些授权的政治和商业意义深感担心，实际上你会经常发现多数工具同时提供 GNU 和 BSD 版本。例如，GNU 的编译器组合（gcc）和 BSD 的 clang 都可以说是顶级的 C 编译器。来自两个阵营的贡献者紧密跟随对方的工作，所以我们可以认为目前存在的差异将会随着时间逐渐消失。

法律解读

美国法律不再提供版权注册系统：除了很少的特例，只要某人写下什么就自然获得了该内容的版权。

发行某个库必然要通过将其从一个硬盘复制到另一个硬盘这样的操作，而且即便带有一点争议，现实中还是存在几种常用机制允许你有权利复制一个有版权的内容。

- GNU 公共许可证：其允许无限制地复制和使用源代码和可执行文件。不过有一个前提：如果你发行一个基于 GPL 许可证的源代码程序或库，你也必须将你的程序的源代码伴随程序发行。注意，如果你是在非商业环境下使用这样的程序，你不需要发行源代码。像用 gcc 编译你的源代码之类的运行 GPL 许可证的程序本身并不会使你具有发行源代码的义务，因为这个程序的数据（比如你编译出的可执行文件）并不认为是基于或派生于 gcc 的。[例

如：GNU 科学计算库。]

- 次级 GPL 许可证：与 GPL 有很多相似，但是具有一个明显的区别：如果你以共享库的方式连接一个 LGPL 库，你的代码不算做派生的代码，也没有发行源代码的义务。也就是说，你可以采用不暴露源代码的方式发行一个与 LGPL 库连接的程序。[例如：Glib。]
- BSD 许可证：要求你在放弃对已带有 BSD 许可部分的版权的同时，保持自己工作的版权，但是不要求你再发行那些 BSD 许可的源代码。[例如 Libxml2，就是带有与 BSD 许可类似的 MIT 许可证。]

注释

本书使用的排版约定

本书使用如下排版约定:

斜体(italic)

用来表示新术语、URL、email 地址、文件名、文件扩展名等。

等宽字体（`Constant width`）

用来表示程序列表，同时在段落中引用的程序元素（例如变量、函数名、数据库、数据类型、环境变量、声明和关键字等）也用该格式表示。

等宽黑体（**`Constant width bold`**）

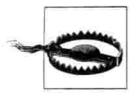
用于表示需要用户逐字符输入的命令或其他文本。

等宽斜体（*`Constant width italic`*）

用于表示应该以用户提供的值或根据上下文决定的值加以替换的文本。

这个图标代表诀窍、建议和一般性的说明。

自己动手：
这里是一个练习，帮助你从实践中学习，并给你一个把手放在键盘上的指标。



这个图标表示警告或错误。

使用示例代码

这是一本试图帮助你解决实际问题的书。总的来说，你可以在你的程序和文档中用本书的代码。除非你复制了太多的部分，你并不需要得到我的许可。例如，你的程序里使用了几段本书的代码并不需要得到许可。但是销售或发行含有 O'Relly 出版书籍中的源代码的确需要许可。通过引用本书及其源代码的方式来回答一个问题不需要得到许可。在你的文档中合并大量来自本书的代码不需要得到许可。

本书中用到的示范代码可以在以下地址找到：http://excamples.oreilly.com/ 063620025108/。

我们感谢，但并不要求，您的引用。一个引用通常包括书名、作者、出版商以及 ISBN 书号。例如，“21 世纪 C 语言，Ben Klemens(O'Reilly)。版权 2013 Ben Klemens, 978-1-449-32714-9。”

如果你感觉你对示范代码的使用可能超出了上面列举的合理情况，你可以随时通过以下邮件地址联系我：permissions@oreilly.com。

网上书店

Safari 网上书店（www.safaribooksonline.com）是一个点播方式的数字图书馆，可以下载世界顶级技术和商业作家的专业书籍和视频。

专业技术人员、软件开发者、Web 设计者、商业和创意人士使用 Safari 网上书店作为他们首选的研究、解决问题、学习和认证培训的信息资源。

Safari 网上书店提供了为组织、政府机关和个人提供了一系列的产品组合和定价套餐。订阅人可以从一个可统一检索的出版商的数据库中找到几千本书籍、培训视频和预发布的手稿，比如 O'Reilly Medial，Prentice Hall Professional、Addison-Wesley Professional、Microsoft Press、Sams、Que、Peachpit Press、Focal Press、Cisco Press、John Wiley & Sons、Syngress、Morgan Kaufmann、IBM Redboks、Packet、Adobe Press、FT Press、Apress、Manning、New Riders、McGraw-Hill、Jones & Bartlett、Course Technology，等等。请登录其网站了解 Safari 网上书店的详情。

如何联系我们

如有对本书的评论或问题，请联系出版商：

美国：

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）

奥莱利技术咨询（北京）有限公司

我们为本书准备了一个网页，其中列举了订正、例子和其他附加信息，地址是：http://oreil.ly /21st_century_c。

如若评论或询问关于本书的技术问题，请将电子邮件发送至：bookquestions@oreilly.com。

关于我们出版的书籍、教材、会议和新闻的进一步的信息，请访问我们的网站：http://www.oreilly.com。

我们在 Facebook 上的地址：http://facebooks/com/oreilly。

请在 Twitter 上关注我们：http://twitter/com/oreillymedia。

请观看我们在 YouTube 上的视频：http://www.youtube.com/oreillymedia。

鸣谢

Nora Albert：慷慨的支持，豚鼠。

Bruce Field、Dave Kitabjian、Sarah Weissman：严谨和彻底的审阅。

Patricj Hall：Unicode 知识。

Nathan Jepson 和 Shawn Wallace：社论。

Rolando Rodriguez：测试、试用和调查。

Rachel Steely：出品。

Ulrik Sverdrup：指出我们可以使用重复指定初始化来设定默认值。

目录

第一部分　开发环境