**简介**

**Mac OS X:构建发展**

复杂的系统以两种方式存在：凭借周密的计划 或者不段更新。机场就是一个事先仔细计划，建造，然后其他的实体轻微的改变即可的样例。复杂的生物体（比如人类）就是一个从简单然后持续进化的样例。最后，适宜进化的有机体总是胜过不适宜的生物进化。

一个操作系统的演化。当然，开发新操作系统的程序员会仔细设计。但最终，一个非常适合当前发展的操作系统将会取代不是如此演化的操作系统。如果是那样的话，那么，一个很有趣的练习是思考是什么特性让操作系统具有进化能力。

最初的Unix版本是由Ken Thompson于1969年在贝尔实验室开发。它被用汇编语言写成运行于一台PDP-7机器。Dennis Ritchie也在贝尔实验室，发明了C语言。在计算机语言中,C是相当低水平的,但它比汇编语言仍有许多便携性。于是Thompson和Ritchie一起用C语言彻底重写了Unix. 1978年，Unix运行在几个不同的体系结构。可移植性，就是，第一个表明Unix适合进化的迹象。

1976年，贝尔实验室开始提供源代码给Unix研究机构。加州大学伯克利分校计算机系统研究小组获得一个副本,开始修改它。Unix的设计非常优雅并且是一个构建两个重要技术的完美平台:虚拟内存和TCP / IP网络。通过自由地分发源代码,贝尔实验室邀请人们来扩展Unix。可扩展性是第二个指示表明Unix是适合进化的。

4.4BSD是 Unix伯克利发布的最后一个版本。它被用作FreeBSD，OpenBSD，NetBSD和Mac OS 系统的基础。今天，Unix被用作于手机和超级计算机的操作系统。它是web服务器、邮件服务器、和工程工作站上最受欢迎的操作系统。 它在很多领域找到了一席之地的方式，又一次表明，Unix能够不段进化。

Mac OS X操作系统是基于 一个混合的Mach 和 4.4BSD系统，但是请注意这是一个新的领域，一个你的祖母会喜欢就像你的孩子喜欢移动设备操作系统一样的桌面操作系统, 是非常不同于Unix系统之前的用意。为达到此目的, 苹果已经在它其 Unix核心添加了几个重要的补充。

Mac OS X中Unix 的一部分叫做Darwin。苹果在Darwin大量增加的部分,已被认可为核心技术。苹果认识到其Unix部分必须继续进化,并已经发布了Darwin和大部分核心技术的源代码。

**这本书**

我的看法是在你专业领域内的事情就不会变得那么“神奇”。我不期望有人来实现一个编译器和操作系统,但是它们不应该是神秘的黑色盒子。

Paul Kim, Noodler首席执行官, Noodlesoft公司

直到我的朋友保罗有一天在IRC上说出了上述一段话，我才意识到这本书和Big Nerd Ranch高级Mac OS X训练营的目的之一，很大程度上是促进阐明Mac OS X基础技术。一个程序员可以只在Cocoa层面呆很长时间，如此一来，在处理NSTask或网络通信时，就只能面对各种变幻莫测的Bug出现。我相信 底层的操作系统知识将帮助您更有效地使用更高级技术。这就是我希望在这里能做到的, 向您展示底层的Unix APIs是如何工作的,然后向您展示高级技术如何使用它们的。

这本书打算在致力于Mac和iOS图形编程方面的高级书籍以及底层内核内部巨著之间架起一座桥梁。一些Unix API你可能已经在大学的操作系统课程看过，因此我会去掉了一些历史记录以及不再相关的细节。例如,你不会看到任何关于管理pseudo-ttys （虚拟终端）的内容。

当你读完这本书,你将能够:

创建充分利用Unix APIs力量的应用程序。

使用像多线程和任务队列的高级思想来提高应用程序的性能。

添加网络功能到事件驱动的应用程序。

使网络应用能够感知Bonjour服务。

使用安全框架的钥匙链和授权功能。

理解和使用gcc，链接器，调试器，Instruments工具，以及Xcode的某个暗角。

使用性能工具来评估和提高你现有应用程序的响应能力。

这本书中的理念可分为三个基本组:

*Unix APIs*

这是一组标准的Unix APIs，每个程序员都应该知道如何使用。虽然高级抽象技术会减少对它们的直接调用，但了解这些功能和结构会给你一个更深入的了解您的系统是如何工作的。这里说的大部分内容也将适用于Linux。

*Framework APIs*

苹果在它的Unix版本中增加了一整套守护进程和框架。这些框架都是极其强大的,有时苹果一直迟迟的文档化他们如何工作以及如何使用。

*Tools*

在Mac OS X开发工具套件中，除了系统的内置工具，提供了一个名副其实的玩具商店给开发者们。这本书涵盖了底层传统Unix工具如gcc，gdb，链接器，DTrace以及高级工具如Instruments。

这本书大部分的代码是ANSI C标准。偶尔有些C99特性将被使用。一些章节使用Cocoa APIs，所以你应该对Cocoa和Objective - C有一个基本的了解 。你可以获得必要的专业知识通过阅读Aaron Hillegass 的Cocoa Programming for Mac OS X前九章内容。iOS开发者可以利用许多操作系统的特性。我也会试图指出Mac OS X和iOS存在的差异。

**排版约定**

为了让这本书更容易理解,我们使用了一些排版约定。

函数名采用粗体等宽字体。所有标准的Unix函数名采用小写字母。苹果公司开发的函数名通常采用是大小写混合。为了使函数更明确，函数名后面将跟随一组括号。例如，您可能会看到，“使用NSLog()或printf()来显示计算值。”。

在Objective-C中，类名称都是大写的。在这本书中，他们也会采用粗体等宽字体。在Objective-C中，方法名称以小写字母开头。他们也将会采用粗体等宽字体。因此，例如，你可能会看到，” NSObject类有一个-dealloc的方法 。”

在代码中的其他文字，将采用在一个常规等宽字体。文件采用相同的字体。因此，您可能会看到“在SomeCode.c中，设置变量foo为nil。”

命令行工具和其他命令将采用一个略小粗体。例如，“用lipo工具合并两个文件，并使用文件命令验证结果。”

偶尔，可能会有一段摘录从一个终端窗口输出。你输的内容也将会采用粗体等宽字体。计算机的响应将会采用常规等宽字体。示例:

**$ ls /var**

at cron empty mail named root tmp yp

backups db log msgs netboot run spool vm

**在线资料**

这本书中的代码可以从http://www.borkware.com/corebook/下载。

此页留白

**致谢**

本书是基于The Big Nerd Ranch教授的*Advanced Mac OS X Bootcamp* 5日课程改写而来。我的学生在好奇心与耐心的驱使下，做了一份完整并且易理解的促使Mac OS X系统稳定，灵活与高性能的管道技术介绍。

特别要感谢Jeremy Sherman。在我没法教课时，Jeremy很快补上并教授了*Advanced Bootcamps其中的一节*课程。一路上他多次改进这本书和相关课程材料,帮助革新和健壮代码。Jeremy还极负责的深入检阅Blocks章节。

尤为感谢Aaron Hillegass,我本书前两个版本的合著者。数月以前，Aaron参加了Asheville的一个Cocoa课程，随意的坐在一个家伙后面。多年来，我从Aaron身上学到的有关写作，讲课以及善待他人比我想象的要多得多。

Susan Loper,不知疲倦的编者，在抒写华丽的散文上表现的令人不可思议。

此页留白

**目录**

前言

介绍