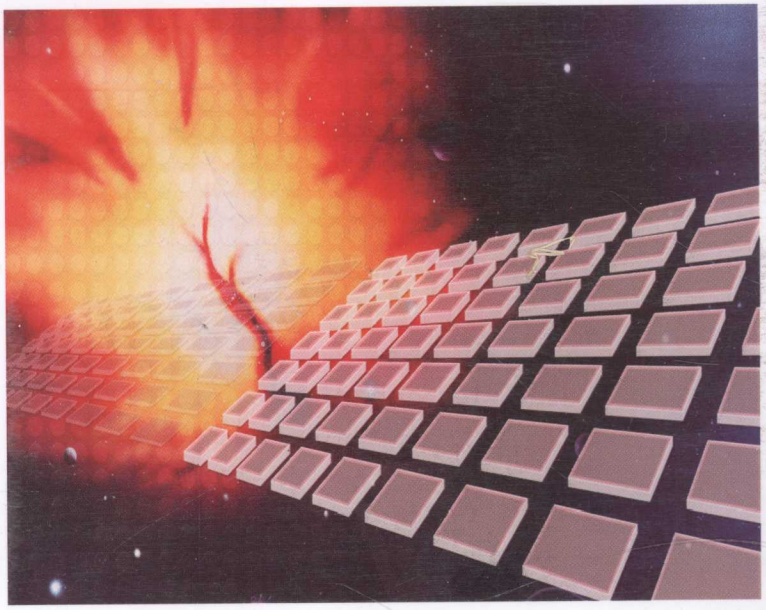
(美) M. Tim Jones 著 张元章 译

CENGAGE Learning™

GNU/LINUX环境编程（第2版）

- 浅显易懂，容易理解的Linux编程基础
- 聚焦Linux编程利器——GNU工具和库
- 全面覆盖各种有用的API
- 配套资源包含本书代码和所有API



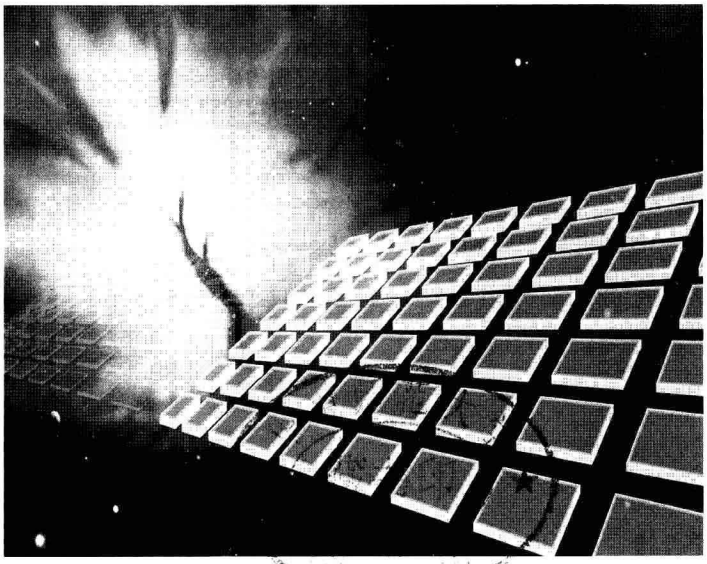
CHARLES RIVER MEDIA

PROGRAMMING SERIES

CENGAGE Learning™ 清华大学出版社

CENGAGE Learning™

GNU/LINUX环境编程（第2版）



清华大学出版社
北　京

CHARLES RIVER MEDIA PROGRAMMING SERIES

内 容 简 介

本书针对 Linux 程序员，本书详细介绍如何开发高性能、高安全性的应用程序。经过全面修订的第 2 版涵括所有必要的工具和编程技术，提供丰富示例来说明 GNU/Linux API 的用法。全书共 5 部分 34 章，主题涉及 GNU 工具，应用程序开发，shell 与脚本编程，调试和提高安全性，虚拟化技术。凡此种种，吸引着近万名程序员选择本书来学习 GNU/Linux 编程技术。

本书浅显易懂，从全局介绍 Linux 编程基础，重点强调 Linux 编程利器的使用，对 Linux 程序员具有重要的参考和指导作用。

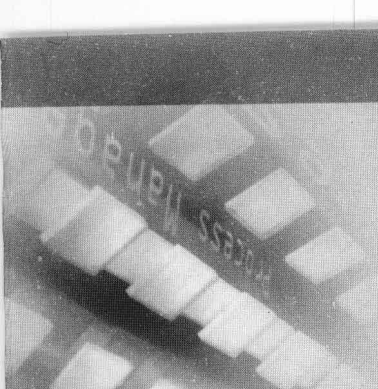
978-7-302-22065-7
Cengage Learning Asia Pte. Ltd.
5 Shenton Way, # 01-01 UIC Building, Singapore 068808

北京市版权局著作权合同登记号　图字：01-2009-6812

图书在版编目(CIP)数据
GNU/LINUX 环境编程(第 2 版)/(美)琼斯(Jones，M. T.)著；张元章译.—北京：清华大学出版社，2010.3
书名原文：GNU/Linux Application Programming，Second Edition
ISBN 978-7-302-22065-7
Ⅰ.G… Ⅱ.①琼… ②张… Ⅲ.Linux 操作系统—程序设计 Ⅳ.TP316.89

中国版本图书馆 CIP 数据核字(2010)第 026222 号

责任编辑：文开琪
责任校对：周剑云
责任印制：孟凡玉
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京密云胶印厂
装 订 者：三河市金元印装有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：34.25　**字　数**：829 千字
版　　次：2010 年 3 月第 1 版　　**印　　次**：2010 年 3 月第1 次印刷
印　　数：1～4000
定　　价：69.00 元



译者序：周虽旧邦，其命维新

张元章

生活在这块土地上的先民们大概不会想象到以后会有这样一个时代，在这个时代中几乎一切都在很快地发生着变化。但先民们确实感觉到，在这个宇宙中，唯一永恒的可能就是变化。

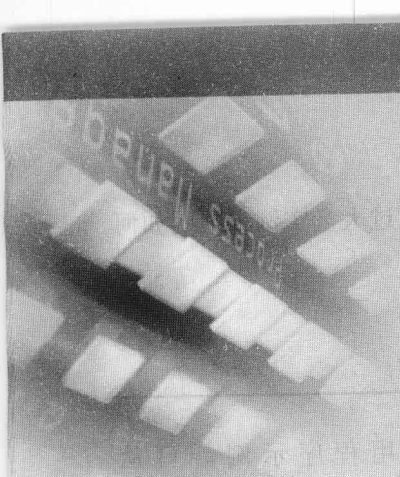
今天，我们都能很明确地体会到自己身处不断地高速度的变化之中。计算机应用程序开发尤其如此，开发工具甚至开发思想都在不断地变化。我们适应着这些变化的同时，也在促成、影响着这些变化。值得欣慰的是，变化的节奏还没有引起大家的不适，而几乎每一次变化都让人类变得更强。

本书展示了用于编写 Linux 应用程序的 GNU 工具和库。重点突出了 Linux 应用程序编程中一些新的变化，如 Ruby 和 Python 不容置疑地兴起；flex 和 bison 的成熟使得构造解析器不再是资深专家的特权；同时本书是帮助你开发 GNU/Linux 操作系统应用程序的利器，即使你原本对 Linux 并不熟悉。本书通过大量的示例演示了各种 API 函数，包括进程管理、共享内存、消息队列、旗语、POSIX、文件操作、套接字等，你可以通过这些示例快速学习 API 函数的使用，快速奠定 GNU/Linux 应用程序编程的基础。

本书在专有名词的使用上，特别地提高了“兼容性”：如“代码硬化”和“防御性编程”，“共享库”和“动态链接库”，这些专有名词在本书中确保不会导致歧义时反复交替使用，表达相同的含义，这是原书本来的特色，以此书入门的读者应该可以容易地适应其他书籍资料中不同的专有名词使用习惯。翻译中保留了这种特色，也交替使用了相同含义的不同中文名词，如“套件”和“软件包”。

最后，我想感谢原作者 M. Tim Jones，是他为我们提供了这样一本出色的教程，本书处处突出“快速”，快速地掌握新工具，快速地面对新变化；同时，也没有忽略对基础理论、基本方法的介绍。这里还要感谢翻译过程中涉及的所有人士，他们是文开琪、史海峰、刘淼、李克武、龙重。

衷心希望读者朋友们能通过本书，体验最新的 GNU/Linux 应用程序编程，加入激动人心的开源社区。



前　言

GNU/Linux 是操作系统中的“瑞士军刀”。从最小的设备(如苹果的 iPod)到最大的设备(如 IBM 的“蓝色基因”巨型机)中都可以看到它的踪影。你会发现 GNU/Linux 运行在很多不同的架构中，从老式的 x86 处理器到 PlayStation 3 使用的 cell 处理器，不一而举。

本书提供 GNU/Linux 操作系统上的应用程序开发的基础知识。无论你是为 iPod 还是 BlueGene 开发应用程序，所需要的编程概念与 API 均可以在本书中找到。

本书内容

本书全面介绍 GNU/Linux 环境下应用程序开发的所有知识。全书五部分，包含 GNU 工具、应用程序开发、shell 与脚本、调试与硬化及一些介绍性的主题，如虚拟化基础。

具体包含以下主题。

- GNU/Linux 架构与虚拟机制。
- GNU 工具，如 GCC、make、automake/autoconf、源代码控制系统、GNU Debugger 以及 GNUplot。
- 应用程序开发基础，如库(静态的和动态的)、文件管理、管道、套接字、编程等。
- GNU/Linux 进程模型(包括线程)和 POSIX IPC 机制(消息序列、旗语及共享内存)。
- shell 与脚本基础，从相应的 GNU/Linux 命令到 Bash、Ruby 以及 Python。sed 和 awk 文本管理，flex 和 bison 语法分析器生成。
- 本书还覆盖调试和硬化技术，包括软件测试工具、覆盖测试和利用 GCov 和 GProf 的剖析及内存调试工具(如 valgrind 等)。

本书配套资源

本书面向的读者

如果你想学习如何开发运行与 GNU/Linux 操作系统有关的应用程序，或者想扩充知识以进入更高深的开发领域，那么本书就是为你而写的，本书适合 GNU/Linux 初级和中级程序开

发人员阅读与参考。书中覆盖相关的工具、API 及开发技术，并通过大量实例来阐明如何使用 GNU/Linux 的 API。

随书资源

随书资源包含本书(GNU/Linux 应用程序编程)所有的应用程序示例。可从 www.wenyuan.com 找到本书后下载，也可通过发送电子邮件到 coo@netease.com 申请和获取。

资源目录

- Source：本书中所有的示例代码，按章保存。
- Figures：本书中所有的示意图，按章保存。

系统需求

- Linux，内核版本 2.6(随书资源经过了 Fedora 和 Ubuntu 的测试)
- 奔腾 I 处理器及以上
- 硬盘驱动器。
- 256MB 以上内存
- 为示例代码准备 1MB 硬盘空间

读者指南

本书是为 GNU/Linux 应用程序开发人员写作的。你会注意到本书没有“Linux 内核”或“设备驱动”这样的主题。本书特意如此，因为尽管那些主题本身也很迷人，但掌握那些内容对于开发 GNU/Linux 环境下的应用程序和工具而言却不是完全必需的。

本书分为五个部分，分别集中讨论 GNU/Linux 编程的不同方面。第 I 部分“导论”，向初学者介绍 GNU/Linux。这部分讲述 GNU/Linux 架构，简要介绍了进程模型和使用许可，简介了开源开发及其使用许可。还探究了 Linux 虚拟化，包括 Linux 中的模型和选项。

第 II 部分“GNU 工具”，集中于 GNU/Linux 编程需要的工具。这部分探究了标准的 GNU 编译工具链和 GNU make 自动 build 系统。研究了库的 build 和用法(静态的和动态的)。探究了使用 gcov 和 gprof 进行覆盖测试和剖析，使用 automake 和 autoconf 进行应用程序打包和发布。最后，回顾 Linux 与源码控制相关的常用选项和 Gnuplot 数据可视化。

在介绍 GNU/Linux 架构和必要的应用程序开发工具的基础上，接下来的第III部分“应用程序开发”集中关注 GNU/Linux 中最有用的服务。包括管道、套接字编程、文件管理、传统的进程和 POSIX 线程、消息队列、旗语 semaphores 和共享内存管理。

在第Ⅳ部分“GNU/Linux shell 和脚本”中，我们转而关注使用 shell 和脚本语言的应用程序开发。在 GNU/Linux 上编程需要用到的一些最有用的命令在这里介绍。第Ⅳ部分还提供一个关于 Bourne-Again shell(bash)的指南。探究了如何使用两种最流行的字符串处理语言(awk 和 sed)来进行文本处理。讨论使用 GNU Flex 和 Bison(lex 和 yacc-兼容分析程序生成器)进行语法分析。这部分还研究了 Ruby 和 Python 脚本。

第Ⅴ部分“调试与测试”从多个不同的方在阐述了调试。我们将研究一些可能有助于自动回归的大型机单元测试。介绍 GNU Debugger 及最常用的命令和技术。最后，探究代码硬化专题及相关的多种调试工具和提升 GNU/Linux 应用程序可靠性和安全性的辅助开发工具。

本书各部分有其内在的线索，但各章也可以根据需要独立阅读。在需要其他章节信息的地方提供了注明。

致谢

1994 年夏天，我第一次接触到“开源”。那时的我刚刚完成一个大型同步通信卫星的操作系统内核开发项目，使用的是 MIL-STD-1750A 微处理器上的 Ada 语言。Ada 语言在技术上十分精确、安全，可读性好。即使以 20 世纪 90 年代早期的标准来看，MIL-STD-1750A 微处理器也是很陈旧的。(这是 20 世纪 70 年代为军用航空电子设备设计的指定集架构，但其简明性仍然是一流的。)

我转而从事伽玛射线暴研究卫星方面的工作，同时从事 1750GALS 项目的验证(validation)工作。Oliver Kellogg 主管的这个项目，包括了针对 1750A 系列的处理器的 Ada 语言的 GCC 编译器、汇编器、连接器和模拟器。我具备 Ada 和 1750A 的相关知识背景，而且伽玛射线暴研究项目当时也进入了试生产阶段，于是我便挪出一些时间进行验证工作。几个月后，我在 comp.compilers 新闻组看到一篇文章，摘录如下：

“1750GALS”，MIL-STD-1750 GCC/汇编器/连接器/模拟器，现在有了欧洲 FTP 站，及一个在美国的 FTP 镜像。

感谢 VTT 公司的 Rekka Ruuska(Pekka.Ruuska@vtt.fi)和 MIT 空间研究所的 M. Tim Jones(mtj@space.mit.edu)，他们的调试报告使这个工具包现在可用。此外，感谢 Tim Jones 架设了美国的 FTP 镜像站。

我无意中变得世界知名且因此而大出风头。当然，这种说法有点夸张，但我帮助那个项目

所花的时间的确既有趣又有价值，我由此进入自由软件(那时自由软件已有 10 年历史)和开放源码(在之后三年，这个术语才被创造出来)的世界。

本书第 2 版不仅仅是数月努力工作的成果，更是根植于 UNIX 和 GNU 工具开发人员们数十年不懈的工作。为创建和改进这些工具做出贡献的不计其数的开发人员的名字足够编成一部专著，所以我在这里简单地向(我认为的)四位对 GNU/Linux 操作系统贡献最突出的人致谢。

AT&T 贝尔实验室的 Dennis Ritchie 和 Ken Thompson，他们创建了第一个 UNIX 操作系统(以及其后继变体)和 C 程序语言。

GNU 和自由软件基金的缔造者 Richard Stallman 激发并团结全世界的自由思想者，创建了世界水平的操作系统 GNU/Linux。

最后但绝不意味着贡献最小，Linus Torvalds 提供了 Linux 内核，而且一直是内核源码的把关者和重要贡献者。

特别感谢 Jim Lieb，他丰富的 UNIX 相关知识及其对本书的详细审阅，使得本书得以很大改进。感谢 Cengage(特别是 Jen Blaney 和 Marta Justak)，他们使第 2 版的出版成为可能。

目　录

第 I 部分　导　论

第Ⅱ部分 GNU 工具

第Ⅲ部分 应用程序开发主题

第Ⅳ部分 GNU/Linux 的 shell 与脚本

第V部分 调试与测试

第Ⅰ部分　导　　论

- 第 1 章　GNU/Linux 的历史
- 第 2 章　GNU/Linux 系统架构
- 第 3 章　自由软件开发
- 第 4 章　Linux 虚拟化与仿真

本书的第Ⅰ部分探究 GNU/Linux 操作系统及其开发范例的几个相关的介绍性主题：UNIX，GNU 及 GNU/Linux 操作系统简史；GNU/Linux 系统架构概述；对自由软件(以及开源问题)开发模式的讨论；简单概述一下 Linux 虚拟与仿真解决方案。

第 1 章　GNU/Linux 的历史

事实上，GNU/Linux 的历史发端于 1969 年第一个 UNIX 操作系统的开发。本章论述 UNIX 的开发历史以及促成 GNU/Linux 操作系统发布的几个重要开发人员所做的积极努力(和所遭受的挫折)。