****

**THE LINUX PROGRAMMING INTERFACE**

**A Linux and UNIX System Programming Handbook**

**MICHAEL KERRISK**

no starch press

San Francisco

*Translated by: Kevin*

***本资料仅供学习所用，请于下载后24小时内删除，否则引起的任何后果均由您自己承担。本书版权归原作者所有，如果您喜欢本书，请购买正版支持作者。***

目录

[前言 7](#_Toc289440689)

[主题 7](#_Toc289440690)

[目标读者 7](#_Toc289440691)

[Linux和UNIX 8](#_Toc289440692)

[使用和组织 8](#_Toc289440693)

[例子程序 9](#_Toc289440694)

[练习 10](#_Toc289440695)

[标准和可移植性 10](#_Toc289440696)

[Linux内核和C库版本 11](#_Toc289440697)

[其它语言使用编程接口 11](#_Toc289440698)

[关于作者 11](#_Toc289440699)

[致谢 11](#_Toc289440700)

[许可 12](#_Toc289440701)

[网站和例子程序源代码 12](#_Toc289440702)

[反馈 12](#_Toc289440703)

[第1章 历史和标准 13](#_Toc289440704)

[1.1 UNIX和C简史 13](#_Toc289440705)

[1.2 Linux简史 16](#_Toc289440706)

[1.2.1 GNU项目 17](#_Toc289440707)

[1.2.2 Linux内核 18](#_Toc289440708)

[第2章 基础概念 23](#_Toc289440709)

[第3章 系统编程概念 24](#_Toc289440710)

[第4章 文件I/O：统一的I/O模型 25](#_Toc289440711)

[第5章 文件I/O：更多细节 26](#_Toc289440712)

[第6章 进程 27](#_Toc289440713)

[第7章 内存分配 28](#_Toc289440714)

[第8章 用户和组 29](#_Toc289440715)

[第9章 进程凭证 30](#_Toc289440716)

[第10章 时间 31](#_Toc289440717)

[第11章 系统限制和选项 32](#_Toc289440718)

[第12章 系统和进程信息 33](#_Toc289440719)

[第13章 文件I/O缓冲 34](#_Toc289440720)

[第14章 文件系统 35](#_Toc289440721)

[第15章 文件属性 36](#_Toc289440722)

[第16章 扩展属性 37](#_Toc289440723)

[第17章 访问控制列表 38](#_Toc289440724)

[第18章 目录和链接 39](#_Toc289440725)

[第19章 监控文件事件 40](#_Toc289440726)

[第20章 信号：基础概念 41](#_Toc289440727)

[第21章 信号：信号处理器 42](#_Toc289440728)

[第22章 信号：高级特性 43](#_Toc289440729)

[第23章 定时器和睡眠 44](#_Toc289440730)

[第24章 进程创建 45](#_Toc289440731)

[第25章 进程结束 46](#_Toc289440732)

[第26章 监控子进程 47](#_Toc289440733)

[第27章 程序执行 48](#_Toc289440734)

[第28章 进程创建和程序执行的更多细节 49](#_Toc289440735)

[第29章 线程：介绍 50](#_Toc289440736)

[第30章 线程：同步 51](#_Toc289440737)

[第31章 线程：线程安全和线程存储 52](#_Toc289440738)

[第32章 线程：线程取消 53](#_Toc289440739)

[第33章 线程：更多细节 54](#_Toc289440740)

[第34章 进程组、会话和任务控制 55](#_Toc289440741)

[第35章 进程优先级和调度 56](#_Toc289440742)

[第36章 进程资源 57](#_Toc289440743)

[第37章 Daemon 58](#_Toc289440744)

[第38章 编写安全的特权程序 59](#_Toc289440745)

[第39章 能力 60](#_Toc289440746)

[第40章 登录会计 61](#_Toc289440747)

[第41章 共享库基础 62](#_Toc289440748)

[第42章 共享库高级特性 63](#_Toc289440749)

[第43章 进程间通信简介 64](#_Toc289440750)

[第44章 管道和FIFO 65](#_Toc289440751)

[第45章 System V IPC介绍 66](#_Toc289440752)

[第46章 System V消息队列 67](#_Toc289440753)

[第47章 System V信号量 68](#_Toc289440754)

[第48章 System V共享内存 69](#_Toc289440755)

[第49章 内存映射 70](#_Toc289440756)

[第50章 虚拟内存操作 71](#_Toc289440757)

[第51章 POSIX IPC介绍 72](#_Toc289440758)

[第52章 POSIX消息队列 73](#_Toc289440759)

[第53章 POSIX信号量 74](#_Toc289440760)

[第54章 POSIX共享内存 75](#_Toc289440761)

[第55章 文件锁 76](#_Toc289440762)

[第56章 Sockets：介绍 77](#_Toc289440763)

[第57章 Sockets：UNIX Domain 78](#_Toc289440764)

[第58章 Sockets：TCP/IP网络基础 79](#_Toc289440765)

[第59章 Sockets：Internet Domain 80](#_Toc289440766)

[第60章 Sockets：服务器设计 81](#_Toc289440767)

[第61章 Sockets：高级主题 82](#_Toc289440768)

[第62章 终端 83](#_Toc289440769)

[第63章 可选I/O模型 84](#_Toc289440770)

[第64章 伪终端 85](#_Toc289440771)

[附录A：跟踪系统调用 86](#_Toc289440772)

[附录B：解析命令行参数 87](#_Toc289440773)

[附录C：转换NULL指针 88](#_Toc289440774)

[附录D：内核配置 89](#_Toc289440775)

[附录E：更多信息来源 90](#_Toc289440776)

[附录F：部分习题解答 91](#_Toc289440777)

[参考书目 92](#_Toc289440778)

[索引 93](#_Toc289440779)

# 前言

## 主题

本书描述Linux编程接口——Linux（UNIX操作系统的一种免费实现）提供的系统调用、库函数、和其它底层接口。这些接口被直接或间接地使用在Linux上运行的每个程序中。它们允许应用程序完成各种任务：如文件I/O、创建删除文件和目录、创建新进程、执行程序、设置定时器、本机进程和线程间通信、通过网络连接的不同机器进程间通信等等。这些底层接口有时候也叫做系统编程接口。

尽管本书关注于Linux，但我也非常注意标准和可移植性问题，清晰地区分了Linux特有的接口、多数UNIX实现共有的特性、以及POSIX和Single UNIX Specification标准定义的特性。因此本书也提供了UNIX/POSIX编程接口的详尽描述，能够适用于编写UNIX系统应用或跨平台应用的程序员。

## 目标读者

本书主要面向以下读者：

* 为Linux、UNIX、或者其它遵循POSIX的系统开发应用的程序员和软件设计师；
* 在Linux、UNIX、或其它操作系统之间移植应用的程序员；
* Linux或UNIX系统编程课程的教师和高年级学生；
* 希望深入理解Linux/UNIX编程接口，以及系统软件是如何实现的系统管理员和“高级用户”。

我假设你拥有一定的编程经验，但不要求系统编程经验。我还假设你了解C编程语言，并且知道如何使用shell和常用的Linux或UNIX命令。如果你是Linux/UNIX的新手，你会发现第2章非常有用，我们以程序员的视角来讲述Linux和UNIX的基础概念。

## Linux和UNIX

本书原本可以纯粹地讲解标准UNIX（也就是POSIX）系统编程，因为UNIX和Linux的大多数特性都是相同的。不过虽然编写可移植程序是很好的目标，理解Linux对标准UNIX编程接口的扩展也是非常重要的。理由之一是Linux非常流行；其二是有时候为了性能、或使用标准UNIX没有的功能，我们不得不使用非标准的扩展（所有UNIX实现都提供类似的非标准扩展）。

因此本书在适用于标准UNIX的程序员时，还提供了Linux特定编程特性的详细描述。这些特性包括：

* epoll，获得文件I/O事件通知的机制；
* inotify，监控文件和目录改变的机制；
* 能力，授予进程一组超级用户能力的机制；
* 扩展属性；
* i-node标志；
* clone()系统调用；
* /proc文件系统
* Linux对文件I/O、信号、定时器、线程、共享库、进程间通信、和socket的特殊实现细节。

## 使用和组织

你至少可以按两种方式来使用本书：

* 作为Linux/UNIX编程接口的介绍手册。你可以从头到尾阅读本书。后续章节建立在之前章节的基础之上，我尽量避免依赖后续章节的情况。
* 作为Linux/UNIX编程接口的索引参考手册。详细的索引和频繁的交叉引用，允许你随机地阅读任何主题。

我把本书分为以下几部分：

1. 背景和概念：UNIX、C和Linux的历史；UNIX标准简介（第1章）；以程序员的视角介绍Linux和UNIX的基本概念（第2章）；Linux和UNIX系统编程的基本概念（第3章）。
2. 系统编程接口的基础特性：文件I/O（第4章和第5章）；进程（第6章）；内存分配（第7章）；用户和组（第8章）；进程凭证（第9章）；定时器（第10章）；系统限制和选项（第11章）；获取系统和进程信息（第12章）。
3. 系统编程接口的高级特性：文件I/O缓冲（第13章）；文件系统（第14章）；文件属性（第15章）；扩展属性（第16章）；访问控制列表（第17章）；目录和链接（第18章）；监控文件事件（第19章）；信号（第20章到第22章）；定时器（第23章）。
4. 进程、程序、和线程：进程创建、进程结束、监控子进程、执行程序（第24章到第28章）；POSIX线程（第29章到第33章）。
5. 进程和程序的高级主题：进程组、会话、任务控制（第34章）；进程优先级和调度（第35章）；进程资源（第36章）；daemon（第37章）；编写安全的特权程序（第38章）；能力（第39章）；登录会计（第40章）；共享库（第41章到第42章）。
6. 进程间通信（IPC）：IPC简介（第43章）；管道和FIFO（第44章）；System V IPC——消息队列、信号量、共享内存（第45章到第48章）；内存映射（第49章）；虚拟内存操作（第50章）；POSIX IPC——消息队列、信号量、共享内存（第51章到第54章）；文件锁（第55章）。
7. Socket和网络编程：IPC和socket网络编程（第56章到第61章）。
8. 高级I/O主题：终端（第62章）；可选I/O模型（第63章）；伪终端（第64章）。

## 例子程序

我用短小但完整的例子程序来阐述多数接口的使用方法，这些例子都被设计为很容易就能从命令行体验，来查看不同的系统调用和库函数如何工作。所以本书包含大量的示例代码——大概15000行C代码和shell会话日志。

尽管阅读和试验例子程序是不错的起点，掌握本书讨论的概念最有效的方法是编写代码，按你的想法修改例子程序，或者编写新程序都可以。

本书的所有源代码都可以在网站上下载。源代码包含许多书中没有的程序。这些程序的目的和细节在注释中都有相关描述。我提供了Makefile编译这些程序，以及一个README文件，给出了例子程序更多的细节信息。

源代码采用GNU Affero 通用公共授权版本3，可以自由分发和修改。源代码中也包含一份该协议的拷贝。

## 练习

多数章节都以一组练习结束，其中一些是要你按不同方式来试验例子程序，另外一些是该章讨论过的概念相关的问题，还有就是要求你来编写代码以巩固你对本书的理解。你可以在附录F找到部分练习的解答。

## 标准和可移植性

贯穿整本书，我都对可移植性问题特别地关注。你会发现很多相关标准的引用，特别是POSIX.1-2001和Single UNIX规范版本3（SUSv3）标准。同时你还将看到这些标准最新修订的细节改变，也就是POSIX.1-2008和SUSv4标准。（由于SUSv3是更大的修订版本，也是本书编写时最广泛有效的UNIX标准，本书讨论的标准大多是SUSv3，并标注出SUSv4不同的地方。除非我明确地提到，你可以假设我们对SUSv3规范的描述也适用于SUSv4）。

对于那些不是标准的特性，我会指出在不同UNIX实现间的差别。我还会突出那些Linux特定的特性，以及Linux与其它UNIX对系统调用和库函数实现上的细小差别。当某个特性我没有明确指出是Linux专有时，你也通常可以假设它在多数或所有UNIX上都有实现。

本书大多数例子程序我都在Solaris、FreeBSD、Mac OS X、Tru64 UNIX、和HP-UX上测试通过（除了那些Linux独有的特性）。为了提高代码在这些系统上的可移植性，本书网站上提供的某些例子程序有一些额外的代码。

## Linux内核和C库版本

本书主要关注Linux 2.6.x系列，这是本书写作时最广泛使用的内核版本。Linux 2.4的某些细节也会提到，我也会指出Linux 2.4和2.6的区别。当Linux 2.6.x系列出现了新特性时（例如2.6.34），我也会特别指出相应的内核版本号。

至于C库，本书则主要关注于GNU C库（glibc）版本2。当然，glibc 2.x系列版本存在差异时，我也会特别指出。

在本书即将印刷时，Linux内核刚刚发布了2.6.35版本，glibc则已经发布2.12版本。本书完全适用于这两个软件版本。Linux内核和glibc将来接口的变化，会在本书的网站上列出。

## 其它语言使用编程接口

尽管例子程序用C语言编写，你也可以在其它编程语言中使用本书讨论的接口——例如编译型语言C++、Pascal、Modula、Ada、FORTRAN、D；解释型语言Perl、Python、Ruby等。（Java则需要采用一种不同的方式JNI）。不同的语言要获取必要的常量定义和函数声明，需要使用不同的技术（C++除外），另外传递函数参数时可能也需要一点额外的工作。此外就没有太大的区别了，核心概念其实都是一样的。因此即使你使用其它的编程语言，你也会发现本书提供的信息是适用的。

## 关于作者

（略）

## 致谢

（略）

## 许可

电子工程学会和开放组织非常友好地许可我引用IEEE Std 1003.1, 2004版本，以及信息技术标准——可移植操作系统接口（POSIX），开放组织基本规范Issue6。完整的标准可以在<http://www.unix.org/version3/online.html>上在线查阅。

## 网站和例子程序源代码

你可以在<http://www.man7.org/tlpi>上找到关于本书更多的信息，包括勘误表和例子程序的源代码。

## 反馈

我非常欢迎代码bug报告、代码改进建议、以及代码可移植性的提高。同样我也欢迎本书的bug报告和改进建议。由于Linux编程接口总是在变化，我也非常高兴能获得关于本书将来版本的改进意见，包括新特性和变化特性。

Michael Timothy Kerrisk

Munich, Germany and Christchurch, New Zealand

August 2010

[mtk@man7.org](mailto:mtk@man7.org)

# 第1章 历史和标准

Linux是UNIX操作系统家族的成员之一。在计算机的术语里，UNIX已经拥有很悠久的历史。第1章的前半部分简述UNIX的历史。我们首先描述UNIX系统和C编程语言的起源，然后讲述导致Linux发展成为今天这个样子的两个关键因素：GNU项目和Linux内核的开发。

UNIX系统最显著的特点之一是它的开发不是被一个厂商或组织控制。相反许多商业和非商业组织都为UNIX的发展做出了贡献。UNIX也因此增加了许多革新的特性，但同时也导致UNIX各个实现之间的分歧越来越大，编写一个能运行于所有UNIX实现的应用也变得非常困难。于是产生了UNIX的标准化运动，我们将在本章后半部分进行讨论。

## 1.1 UNIX和C简史

第一个UNIX由贝尔实验室（电话公司AT&T的一个部门）的Ken Thompson在1969年开发完成（Linus Torvalds也正是在这一年出生）。这个UNIX是用汇编为Digital PDP-7微计算机编写。UNIX这个名字和MULTICS（Multiplexed Information and Computing Service）有关，后者是AT&T与麻省理工学院（MIT）和通用电子之前合作开发的操作系统项目。（由于该项目最初的失败，没有能够开发出一个有用的系统，当时AT&T已经退出项目）。Thompson的新操作系统从MULTICS中借用了一些设计，包括树型结构文件系统、对命令解释执行采用独立的程序（shell）、以及把文件当作无结构的字节流。

在1970年， UNIX使用汇编语言为新的Digital PDP-11微计算机重新编写，这个PDP-11的遗留痕迹至今仍然可以在多数UNIX实现中找到，包括Linux。

不久之后，Dennis Ritchie，Thompson在贝尔实验室的一个同事，设计和实现了C编程语言。这是一个进化的过程，C起源于更早的解释语言B，最初由Thompson实现了B语言，并从一个更早的语言BCPL中借鉴了许多想法。到1973年，C已经成熟到UNIX内核几乎可以全部使用其重写。UNIX也因此成为最早使用高级语言编写的操作系统，使其迁移到其它硬件体系架构成为可能的重要因素。

C语言的这个起源，解释了C和C++成为今天最广泛的系统编程语言的原因。之前广泛使用的语言都是为其它目的而设计的：FORTRAN为工程师和科学家完成数学任务；COBOL为商业系统处理面向记录的数据流。C填补了一个空白，和FORTRAN、COBOL不一样的是，C语言是几个人为了一个目标而设计的：开发一个高级语言来实现UNIX内核和相关的软件。和UNIX操作系统本身一样，C由专业的程序员为自身所设计。所产生的语言是小巧、高效、强大、简洁、模块化、注重实效、和一致的。

UNIX第一至第六版

在1969年到1979年间，UNIX发布了一系列版本。本质上就是AT&T对UNIX开发进展的一个快照。UNIX最初的六个版本发布时间如下：

* 第一版，1971年11月：此时UNIX还运行在PDP-11上，已经拥有一个FORTRAN编译器，和许多今天依然在使用的工具，包括ar, cat, chmod, chown, cp, dc, ed, find, ln, ls, mail, mkdir, mv, rm, sh, su, who。
* 第二版，1972年6月：UNIX安装在AT&T内部的10台机器上。
* 第三版，1973年2月：这个版本包含一个C编译器和管道的最初实现。
* 第四版，1973年11月：第一个几乎全部用C编写的版本。
* 第五版，1974年6月：此时UNIX已经安装在超过50个系统中。
* 第六版，1975年5月：这是第一个在AT&T范围外广泛使用的版本。

在这些版本发布的过程中，UNIX的使用和声望得到了扩展，首先在AT&T内部，随后在外部。Communications of the ACM杂志发表的一篇关于UNIX的论文也为此做出了巨大贡献。

当时AT&T正在接受美国电话系统对其垄断的政府制裁。AT&T与美国政府的协议禁止其销售软件，这也意味着AT&T不能把UNIX作为产品销售。相反，从1974年的第五版开始，特别是第六版，AT&T授权大学免费使用UNIX。针对大学的UNIX发布版包含文档和内核源代码（当时大约10000行）。

AT&T对大学发布UNIX极大地促进了UNIX的使用和流行，到1977年UNIX已经运行在500个地方，包括125所美国大学和其它一些国家。当时的商业操作系统非常昂贵，而UNIX为大学提供了一个交互式多用户的操作系统，即便宜又强大。同时UNIX还给大学计算机科学研究提供UNIX操作系统的源代码，他们可以修改并提供给学生学习和体验。很多学生学习了UNIX之后，就成为了UNIX的布道者。其它则加入或组建自己的公司，销售运行着UNIX操作系统的计算机工作站。

**BSD和System V的诞生**

1979年1月UNIX发布了第七版，改进了系统的可靠性，提供了一个增强的文件系统。这个发布版还包含一些新的工具，包括：awk, make, sed, tar, uucp, Bourne shell, 和FORTRAN 77编译器。第七版的发布对于UNIX来说具有重要意义，因为从这一刻起，UNIX产生了两个重要的变种：BSD和System V，它们的起源我们马上就会简要地描述。

Thompson在1975/1976学年回到自己的母校，加州大学伯克利分校担任客座教授。在那里他和几个毕业生为UNIX增加了许多新特性。（其中一个学生Bill Joy，随后与别人一起组建了Sun Microsystems，成为UNIX工作站市场早期参与者）。Berkeley开发了许多新的工具和特性，包括C shell、vi编辑器、改进的文件系统（Berkeley Fast File System）、sendmail、Pascal编译器、新的Digital VAX体系架构下的虚拟内存管理等。

在Berkeley Software Distribution（BSD）的授权许可下，这个版本的UNIX，包括它的源代码，被广泛地发布出去。1979年发布了第一个完整发行版3BSD（更早的Berkeley-BSD和2BSD，只是增加Berkeley开发的新工具，而不是完整的UNIX发行版）。

到1983，加州大学伯克利的计算机系统研究组织（Computer Systems Research Group）发布了4.2BSD。这是一个重大的发行版，因为它包含了完整的TCP/IP实现，包括socket应用编程接口（API）和许多网络工具。4.2BSD和它的前任4.1BSD被广泛发布于全世界的许多大学。它们也构成了Sun公司的UNIX变种，SunOS（1983首次发布）的基础。其它重要的BSD发布包括1986年的4.3BSD，以及1993年的最终发布版：4.4BSD。

与此同时，US反托拉斯诉讼强制AT&T解散（法律诉讼起于1970年代中期，1982年解散生效），由于在电话系统中不再垄断，公司被允许运营UNIX。结果就是1981年System III的诞生。AT&T的UNIX支持组（USG）负责开发System III，它雇佣了数百名开发者来增强UNIX，和开发UNIX应用（著名的有document preparation package和软件开发工具）。随后在1983年发布了System V(5)的第一个版本，一系列的小发布版后最终是1989年的System V发布版4（SVR4），到这时System V已经吸收了BSD的许多特性，包括网络基础设施。System V授权给许多商业厂商，这些厂商使用System V作为自己UNIX实现的基础。

因此到1980年代末，除了各种BSD发布版在大学广泛使用，UNIX还在许多硬件上拥有各种商业实现：包括Sun的SunOS及随后的Solaris、Digital的Ultrix和OSF/1（经过一系列的改名和收购之后，成为了今天的HP Tru64 UNIX）、IBM的AIX、Hewlett-Packard（HP）的HP-UX、NeXT的NeXTStep、Apple Macintosh的A/UX、Microsoft和SCO为Intel x86-32体系架构开发的XENIX。（本书将Linux的x86-32实现统一称为Linux/x86-32）。这种状况和当时典型的私有硬件/操作系统的方式完全不同，后者通常是厂商只生产一个或少数私有计算机芯片体系架构，然后在上面销售自己的私有操作系统。多数厂商系统的这种私有属性，意味着购买受限于一个厂商。切换到另一种私有操作系统和硬件平台会非常昂贵，因为需要迁移现有应用并进行相关的重新训练。这个因素再加上各个厂商便宜的单用户UNIX工作站，使得可移植的UNIX系统对商业应用非常具有吸引力。

## 1.2 Linux简史

Linux这个术语通常引用基于Linux内核的完整的类UNIX操作系统。不过这是错误的叫法，因为典型商业Linux发行版的许多关键组件，都起源于另一个项目，这个项目比Linux要早好几年。

### 1.2.1 GNU项目

Richard Stallman是一个天才程序员，曾工作于MIT，他在1984年开始考虑实现一个"Free" UNIX。Stallman对"free"的观点是精神上的自由，并且定义在法律层面上，而不仅仅是免费（参考<http://www.gnu.org/philosophy/free-sw.html>）。无论如何，Stallman倡导的自由也就意味着软件（如操作系统）应该免费或非常便宜。

Stallman大大影响了厂商对私有操作系统系统附加的限制。这些限制意味着购买计算机软件通常不包含源代码，而且通常不能对该软件进行复制、修改、和分发。Stallman指出这种形式鼓励程序员互相竞争并且保密自己的工作，而不是互相合作和共享成果。

于是Stallman创建了GNU项目（GNU's not UNIX），目标是开发一个完整、自由、类UNIX的系统，包含一个内核和所有相关的软件包，并且鼓励其它人参与该项目。到1985年，Stallman成立了自由软件基金会（FSF），这是一个旨在支持GNU项目以及其它自由软件开发的非赢利组织。

GNU项目的一个重要成果就是GNU General Public License(GPL)的产生，这也是Stallman对自由软件精神的具体化。Linux发行版的多数软件，包括内核都按GPL（或者类似的许可）授权。GPL授权的软件必须使源代码自由可用，而且允许按GPL许可自由地重新发布。GPL授权的软件允许自由地修改，但是修改后的软件必须同样遵循GPL许可。如果修改后的软件以可执行方式发布，作者必须同时允许以不超过发布的代价获得修改过的源代码。GPL第一版发布于1989年，目前的版本3发布于2007年。版本2发布于1991年，目前使用最广泛，也是Linux内核采用的授权。

GNU项目最初并没有开发出一个可用的UNIX内核，但确实创建了许多其它程序。由于这些程序设计成在类UNIX操作系统中运行，它们可以也确实被用在现有的UNIX实现中，有些还迁移到其它操作系统。GNU项目最著名的程序有Emacs文本编辑器、GCC（最早是GNU C编译器，不过现在重新命名为GNU编译器集合，包含C、C++和其它语言的编译器）、Bash shell、和glibc（GNU C库）。

在1990年代初期，GNU项目已经拥有了一个几乎完整的系统，除了一个关键的组成：可用的UNIX内核。GNU项目开始规划一个野心勃勃的内核设计，被称为GNU/HURD，基于Mach微内核。不过HURD远远达不到可发布的程度。（在本书写作之时，HURD的工作仍在继续，目前只能运行在x86-32体系架构下）。

万事俱备，只欠东风。GNU项目已经创建了完整UNIX系统所需的一切，只差一个最重要的内核了。

### 1.2.2 Linux内核

Linus Torvalds在1991年还是芬兰郝尔辛基大学的一名学生，当时他想为自己的Intel 80386 PC编写一个操作系统。在Linus的课程学习过程中，他接触了Minix，由Andrew Tanenbaum在1985年左右开发的类UNIX操作系统内核，后者是荷兰某大学的教授。Tanenbaum创造了Minix，并提供完整的源代码，用作大学操作系统设计课程的教学工具使用。Minix内核可以在386系统中构建和运行，但是由于主要目的是教学工具，Minix设计成很大程度上独立于硬件体系架构，因此不能完全发挥386处理器的能力。

于是Torvalds启动了自己的项目，开始为386创建一个高效、全功能的UNIX内核。几个月之后，Torvalds开发了一个基本的内核，允许自己编译和运行许多GNU程序。然后在1991年10月5日，Trovalds开始在网上请求其它程序员的帮助，发出了下面这段被广泛引用的声明，他在comp.os.minix Usenet新闻组上发布了自己内核的0.02版：

你是否怀念 minix-1.1 版时的日子？那时人们干劲十足，自己编写设备驱动程序。你是否手头正缺少一个很好的项目，并且非常渴望为符合自己的需要动手修改一个操作系统？当几乎所有的程序都能在Minix上运行时，你是否感到非常失望？不再有为了调通一个巧妙的程序而整夜不睡觉的夜猫子？那么本消息（邮件、公告）可能正是为你而发布的:-)。

正如我一个月前所提到的，我正在开发一个用于 AT-386 微机类似于Minix 的操作系统。它目前已经达到了可用的程度(当然，能不能用还依赖于你的具体要求)，而且我很高兴把源代码拿出来广泛发布。目前它的版本是0.02(加上已经编制好的(很小的)补丁程序，就是0.03)，但是我已经在它上面成功地运行了 bash/gcc/gnu-make/gnu-sed/压缩程序等。

该小巧项目的源程序可以在nic.funet.fi(128.214.6.100)上/pub/OS/Linux 目录中找到。该目录中含有一些README文件以及几个在Linux下运行的二进制执行程序(bash, update和gcc，你还能要求什么呢:-)。提供了完整的内核源代码，而且没有使用minix的代码。库文件的源代码仅是部分免费的，所以目前不能给出。照内核现在的样子，系统已经可以进行编译，并且已经可以运行。二进制执行程序（bash和gcc）的源代码可以在同一个地方的/pub/gnu目录中找到。

当心！警告！注意！这些源代码仍然需要minix-386系统来进行编译（需要 gcc-1.40，1.37.1可能也能用，但没有试过），并且如果你想运行它的话还需要minix来进行设置，所以对没有minix的人来说，它至今它还不是一个独立的系统，不过我正在朝这方面努力着。你还需要有些骇客的本事来设置它，所以对那些希望一个minix-386取代品的人来说，就不用考虑Linux了。它目前主要是供对操作系统感兴趣的骇客使用的，并且有能使用minix的386机器。该系统需要一个AT兼容硬盘（IDE硬盘当然更好）以及EGA/VGA显示卡，如果你还感兴趣的话，就使用ftp下载README/RELNOTES文件看看，并且/或者给我EMAIL告之其它信息。

我能够（当然，几乎是）听到你问自己“为什么？”，Hurd将在近年（或者两年、或者下个月，谁知道）内推出，而且我已经有了minix。这是一个骇客为骇客们写的程序，在开发过程中我已经得到了快乐，而某些人可能也乐意阅读它，甚至为自己的需要而修改它。它仍然很小，足以理解、使用和修改，我正期望你可能有的任何建议和说明。 我也对为minix系统编写过工具软件/库函数的任何人的反馈信息感兴趣。如果你的软件是可以自由发布的(在版权下甚至公共域内)，那么我很希望得到你们的消息，这样我就可以将它们加入到Linux系统中。现在我正使用着Earl Chews的stdio（Earl，谢谢你的很好而又能使用的系统），很欢迎这种类似的软件。你的版权当然会保留着，如果你乐意我使用你的代码，就请告知。

Linus

按照传统UNIX克隆采用的X字母结尾命名惯例，这个内核最终命名为Linux。最初Linux采用更加受限制的授权，不过Torvalds很快就将Linux许可更换为GNU GPL协议。

Linus的请求帮助得到热烈影响。很多程序员加入Linux的开发，添加了许多特性，例如增强的文件系统、网络支持、设备驱动、和多处理器支持等。到1994年3月，开发者们发布了1.0版本，1995年3月发布了Linux 1.2，1996年6月发布了Linux 2.0，1999年1月发布了Linux 2.2，2001年1月发布了Linux 2.4。2001年11月开始内核2.5的开发，到2003年12月发布了Linux 2.6。

**BSD**

值得一提的是1990年代前期，另一个免费的UNIX也已经能够用于x86-32体系架构。Bill和Lynne Jolitz对一个已经很成熟的BSD系统向x86-32做了迁移，名叫386/BSD。迁移基于BSD Net/2发布版（1991年6月），是4.3BSD的一个版本，把所有AT&T私有的源代码都替换或移除掉。Jolitz夫妇把Net/2迁移到x86-32，并重写了缺失的代码，在1992年2月发布了386/BSD的第一个版本（V0.0）。

在经历了最初短暂的成功和流行之后，386/BSD的工作由于各种原因而停滞。随着大量patch逐渐积压得不到处理，两个开发团队应运而生，分别创建了自己基于386/BSD的发布版：NetBSD，强调在各种硬件之间保持可移植性；FreeBSD，强调性能，也是现代BSD中最流行的一个。NetBSD的第一个发布版是1993年4月的0.8；FreeBSD的首张CD-ROM（版本1.0）发布于1993年12月。另外还有一个OpenBSD，派生自NetBSD项目，在1996年发布了最初的2.0版本，OpenBSD特别强调安全性。到2003年中期，一个新的DragonFly BSD又从FreeBSD 4.x分离而出。DragonFly BSD采用了不同于FreeBSD 5.x的方式，特别为对称多处理器（SMP）体系架构设计。

如果不提到UNIX系统实验室（USL，负责开发和销售UNIX的AT&T子公司）和伯克利之间的诉讼，那我们对于BSD的讨论就不是完整的。在1992年初，合并成立了伯克利软件设计公司（BSDi，今天是Wind River的一部分），开始发布一个商业支持的BSD UNIX：BSD/OS，基于Net/2发行版和Jolitz夫妇的386/BSD增强功能。BSDi以995美元发布二进制和源代码，并且建议潜在客户使用他们的电话号码1-800-ITS-UNIX。

1992年4月，USL向BSDi正式提出诉讼，试图阻止BSDi销售包含USL私有源代码和商业秘密的产品。USL同时还要求BSDi停止使用迷惑性的电话号码。这个官司最终扩大为要求加州大学赔偿。法院最后判决同意了USL的两个主张，并驳回了其它请求。接着马上加州大学向USL提出反诉讼，声称USL未经许可在System V中使用了BSD代码。

官司正在悬而未决的时候，Novell收购了USL，其CEO（Ray Noorda）开始公开声明自己希望双方在市场上而不是法院里竞争。诉讼最终得以在1994年1月终结，加州大学必须移除Net/2发布版18000个文件中的3个，并对其它少数文件做一些很小的修改，另外还要对大约70个文件增加USL版本声明，而且这些文件不能够再次发布。这个修改后的系统在1994年6月发布为4.4BSD-Lite（加州大学发布的最后一个版本是1995年6月的4.4BSD-Lite版本2）。从这时开始，法律条款要求BSDi、FreeBSD、NetBSD用修改后的4.4BSD-Lite源代码替换Net/2。尽管这导致BSD派生开发的一定延迟，但也使这些系统通过三年的开发，从加州大学计算机系统研究组织发布Net/2后重新同步到一起。

**Linux内核版本号**

和多数自由软件项目一样，Linux采用尽早发布、经常发布的模型，因此新的内核修订频繁更新（有时候几乎每天）。随着Linux用户群的增长，对发布模型进行了一定的修改，以减少对现有用户的影响。具体来说，从Linux 1.0发布之后，内核开发者就采用了固定的内核版本命名规范，每个发布版本统一命名为x.y.z：其中x表示主版本号；y表示在该主版本号下的副版本号；而z则是副版本号下的修订版本号（通常是很小的改进和bug修复）。

在这样一种模型下，通常会有两个内核版本总是处在开发过程中：一个是稳定版，用于生产系统，其主版本号为偶数；另一个是开发版，相对来说不稳定一些，主版本号一般是下一个奇数。理论上（实践中并不总是）所有新特性都只应该添加在当前开发版内核中，而稳定版的修订系列严格限制为很小的改进和bug修复。当内核开发者认为开发版本适合发布时，就会成为新的稳定版，并赋予一个偶数版本号。例如2.3.z开发内核最终形成了2.4稳定内核版本。

2.6内核发布之后，开发模型发生了变化，主要目的是解决稳定版内核发布时间间隔太长导致的问题和挫折（Linux 2.4.0和2.6.0之间差不多有三年时间）。关于改善开发模型的谈论时不时都有进行，但是核心细节基本保持如下：

* 不再有稳定和开发版的明确区分。每个新的2.6.z发布都可以包含新特性，而且都经历增加新特性，然后通过几个候选发布版达到稳定的生命周期。当候选版本足够稳定时，就发布为内核2.6.z版本。发布周期大约三个月。
* 有时候稳定的2.6.z发布版需要小的patch来修复bug或安全性问题。如果这些修复有足够高的优先级，而且这些patch也足够简单到不可能出错，那么不需要等待下一个2.6.z发布版，可以直接创建一个2.6.z.r发布版，这里的r序列号表示2.6.z内核的副修订版本。
* 额外的责任被转移到发行版厂商，来确保发行版内核的稳定性。

后面章节有时候遇到特殊的API时，会提及具体的内核版本（例如新的或修改的系统调用）。不过在2.6.z系列内核之前，多数内核变更都发生在奇数开发版中，我们通常会注明这个变化是在下一个稳定版中产生的，因为多数应用开发者都是使用稳定版内核而不是开发版内核。许多情况下，手册页则会精确地标注某个特性是在哪个开发版出现或修改的。

对于2.6.z系列内核出现的变化，我们会标注具体的内核版本号。当我们说某个特性是内核2.6的新特性时，如果不带z修订号，就表示这个特性是在2.5开发内核中实现的，首次出现在稳定内核版本2.6.0。

**移植到其它硬件体系架构**

# 第2章 基础概念

# 第3章 系统编程概念

# 第4章 文件I/O：统一的I/O模型

# 第5章 文件I/O：更多细节

# 第6章 进程

# 第7章 内存分配

# 第8章 用户和组

# 第9章 进程凭证

# 第10章 时间

# 第11章 系统限制和选项

# 第12章 系统和进程信息

# 第13章 文件I/O缓冲

# 第14章 文件系统

# 第15章 文件属性

# 第16章 扩展属性

# 第17章 访问控制列表

# 第18章 目录和链接

# 第19章 监控文件事件

# 第20章 信号：基础概念

# 第21章 信号：信号处理器

# 第22章 信号：高级特性

# 第23章 定时器和睡眠

# 第24章 进程创建

# 第25章 进程结束

# 第26章 监控子进程

# 第27章 程序执行

# 第28章 进程创建和程序执行的更多细节

# 第29章 线程：介绍

# 第30章 线程：同步

# 第31章 线程：线程安全和线程存储

# 第32章 线程：线程取消

# 第33章 线程：更多细节

# 第34章 进程组、会话和任务控制

# 第35章 进程优先级和调度

# 第36章 进程资源

# 第37章 Daemon

# 第38章 编写安全的特权程序

# 第39章 能力

# 第40章 登录会计

# 第41章 共享库基础

# 第42章 共享库高级特性

# 第43章 进程间通信简介

# 第44章 管道和FIFO

# 第45章 System V IPC介绍

# 第46章 System V消息队列

# 第47章 System V信号量

# 第48章 System V共享内存

# 第49章 内存映射

# 第50章 虚拟内存操作

# 第51章 POSIX IPC介绍

# 第52章 POSIX消息队列

# 第53章 POSIX信号量

# 第54章 POSIX共享内存

# 第55章 文件锁

# 第56章 Sockets：介绍

# 第57章 Sockets：UNIX Domain

# 第58章 Sockets：TCP/IP网络基础

# 第59章 Sockets：Internet Domain

# 第60章 Sockets：服务器设计

# 第61章 Sockets：高级主题

# 第62章 终端

# 第63章 可选I/O模型

# 第64章 伪终端

# 附录A：跟踪系统调用

# 附录B：解析命令行参数

# 附录C：转换NULL指针

# 附录D：内核配置

# 附录E：更多信息来源

# 附录F：部分习题解答

# 参考书目

# 索引