THE LINUX PROGRAMMING INTERFACE

A Linux and UNIX® System Programming Handbook

MICHAEL KERRISK

no starch press

THE LINUX PROGRAMMING INTERFACE

A Linux and UNIX System Programming Handbook

MICHAEL KERRISK

no starch press

San Francisco

Translated by: Kevin

目录

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

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