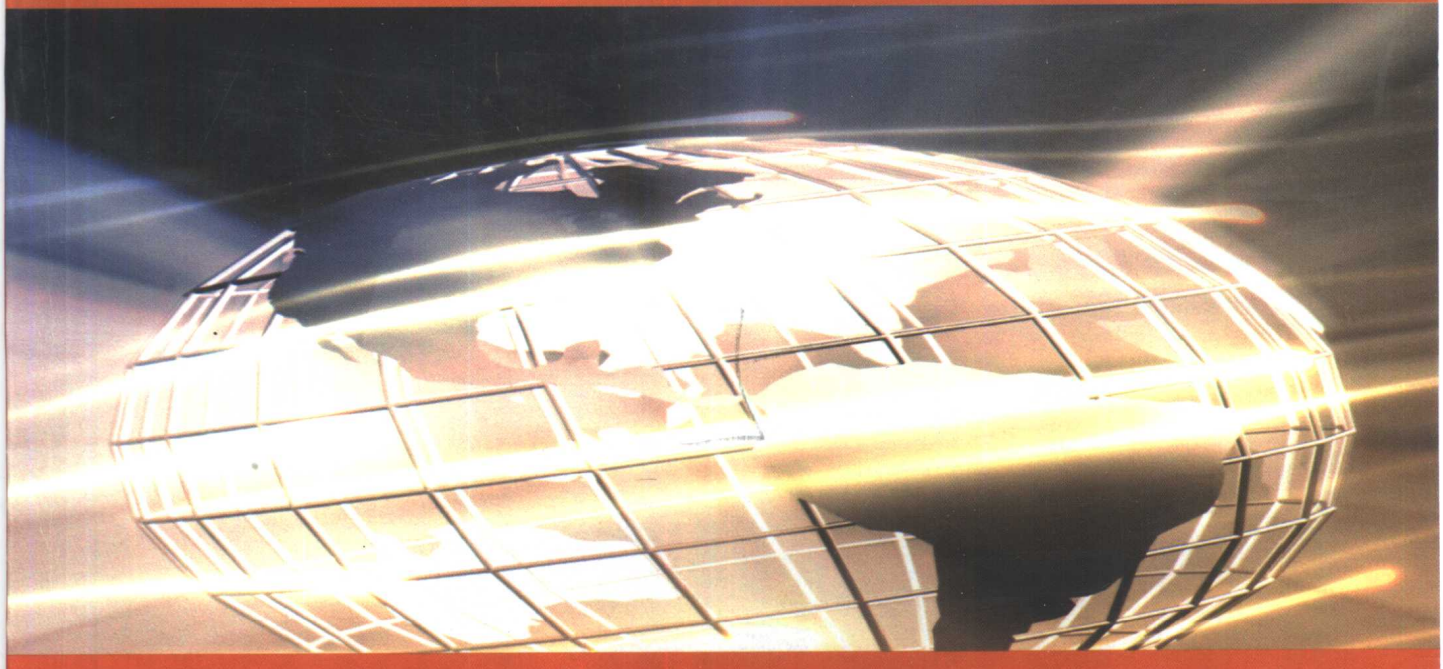
开源软件丛书

Linux Application Development

Linux编程权威指南



Michael K.Johnson
Erik W.Troan 著

龙华乔 胡以迄 译

Addison-Wesley

中国电力出版社
www.infopower.com.cn

内 容 提 要

本书由来自 Red Hat 公司的资深程序师撰写。全书分为四个部分。第一部分介绍操作系统协议术语、文件和运行环境；第二部分介绍编译、链接、程序输入和调试工具；第三部分介绍系统内核和程序库界面；第四部分介绍综合应用。本书几乎涵盖了 Linux 编程的各个层面，内容由浅入深，可读性较高。

本书适合软件设计开发人员及大专院校师生阅读。

图书在版编目（CIP）数据

Linux 编程权威指南/（美）强森等编著；龙华乔等译.-北京：中国电力出版社，2001

ISBN 7-5083-0626-0

Ⅰ.L… Ⅱ.①强…②龙… Ⅲ.Linux 操作系统 Ⅳ.TP316.89

中国版本图书馆 CIP 数据核字（2001）第 25890 号

北京版权局著作权登记号 **图字** 01-2000-3093

本书英文版原名：Linux Application Development

Published by arrangement with Addison Wesley Longman, Inc.

中国电力出版社出版、发行

（北京三里河路 6 号 100044 http://www.infopower.com.cn）

北京市地矿印刷厂

各地新华书店经售

2001 年 7 月第一版 2001 年 7 月北京第一次印刷

787 毫米×1092 毫米 16 开本 23.5 印张 528 千字

定价 45.00 元

序　　言

这本书是为那些具有编程经验（或虽然没有经验，但有兴趣进行学习）的程序员而编写的，以指导他们开发 Linux 软件或者把软件从其他平台移植到 Linux 系统上。这本书是我们自己在学习 Linux 环境下编程时就想拥有的，而且现在我们就把这本书摆在桌上作为参考。在我们着手编写头三章的时候，我们已经用草稿作为工作时的参考材料了。

Linux 设计得很像 Unix。这本书在 Unix 的编程基础和风格方面会向你展现一个清晰的框架背景。Linux 与 Unix 并不是根本不同的——它们的相异之处只在于，如果你是一位平时只依赖 Unix 编程参考书来编程的程序员，而那本参考书又完全不提及 Linux 的话，那么 Linux 会令你伤透脑筋。因此，这本书的确是一本以 Linux 观点来写的 Unix 编程指南。

Linux 有它本身独特的扩展部分，例如它的直接屏幕访问功能（参见第 20 章），而且 Linux 也有一些应用得比其他系统多的功能部件，例如 S-Lang 程序库（参见第 22 章）。这本书的内容涵盖了许多 Linux 独有的扩展部分和功能部件，以便你能真正利用 Linux 的优点来编写程序。

- 如果你是一个熟悉 C 语言的程序员，但不懂 Unix 和 Linux，那么只要你逐页仔细地学习本书，并按照书中的例子进行编程，你将成为一个出色的 Linux 程序员。借助一些其他的专用系统文档，你会发现过渡到 Unix 的任意版本是很容易的。

- 如果你是一个熟练的 Unix 程序员，你将发现这本书会帮助你更容易地过渡到 Linux。我们已经尽了最大的努力，使你能容易地在这本书中准确获取你所需要的知识。我们也仔细并且清晰地论述一些复杂的主题，这些主题有时候甚至连有经验的 Unix 程序员都感到头痛，例如进程和会话群、作业控制和 tty 处理。

- 如果你已经是一个 Linux 程序员，由于本书清晰地论述了一些令人困惑的问题，因此它会使你的许多编程任务变得更轻松。此时几乎每一章你都可以独立进行阅读，因为你已经掌握了 Linux 最低限度的基础知识。无论你如何有经验，都会发现近在手边的这本书是你很需要的。

这本书与普通的 Unix 编程书籍不同，因为它是在专门论述一个特殊的操作系统。我们没有必要迷惑新学者，说什么“BSD 是这样的，SVR4 是那样的，HPUX 有它自己的方式这样做，而 SGI 也有自己的方式。”我们将具体论述每一种情形，然后让你把它们全部整理出来。凭我们自己的经验，可以讲只要你掌握了在任意二种 Unix 系统上熟练编程，那么在别的系统上编程都会变得容易起来。

这本书不想把 Linux 编程的所有细节都覆盖进去。例如，它没有论述 X Windows 系统编程的问题，因为其实这些编程与在任何 Unix 或 Linux 平台上编程都是完全一样的。另外，这本书也没有解释由 ANSI C 所指定的基本界面问题——别的书籍已经解释得非常详细。我们简练地论述了从一个 Dos、Windows 或 Macintosh 等系统下的 C 语言程序员，过渡为一个

Linux 系统下的 C 语言程序员所必需的信息。我们没有论述可以在 Linux 系统下使用的编程语言的功能，而且也没有论及在任何支持系统下表现效果相同的图形编程库。相反，我们为你指明了专门论述上述领域的书籍。

本书分四部分来论述。

- 第一部分向你介绍 Linux——操作系统协议术语，文件和运行环境。
- 第二部分论述了开发环境中最重要的方面——编译、链接、程序输入和一些在别的平台上不常用的调试工具。
- 第三部分是这本书的核心——它描述了连接内核和系统程序库的界面，主要是连接内核的界面。只有这部分的最后三章是 Linux 所特有的，这部分的大部分内容都以透视 Linux 的角度去论述一般的 Unix 编程。
- 第四部分对你所学的知识进行综合——包括描述了一些重要的能提供独立于内核的界面的程序库。准确来讲，这些程序库并不是 Linux 所特有的，但有几个在 Linux 系统中较为常用。

如果你对 Linux 或 Unix 编程已经熟悉了，你可以以任意的次序去学习本书的各章节，不要强迫自己去看那些你不感兴趣的内容。如果你对 Linux 或 Unix 不熟悉，那么大部分的章节都将是独立的，但你也很可能想首先参见第 1、2、4、5、8、9、10 和 11 章，因为这几章向你提供了大部分读其他章节所必需的基础知识。特别是第 9、10 和 11 章，它们构成了 Unix 和 Linux 编程模块的核心。

下面的这几本书，虽然在某些地方可能会有重复，但对这本书作了更简单或更高级的补充，或讲述了相关的主题。

- 《The C Programming Language》[Kernighan，1988] 简明的讲授了 ANSI 标准 C 语言编程，但缺乏操作系统参考。他推荐读者具有一些编程知识或“接近一个更有知识的同事”。
- 《Practical C Programming》[Oualline，1993] 逐步传授 C 语言编程和风格，针对那些没有多少编程经验的人而采用了易学的方法。（译注：本书中文版《实用 C 语言》已由中国电力出版社出版）
- 《Programming With GNU Software》[Loukides，1997] 介绍 GNU 编程环境，包括运行 C 编译器，调试器，制造工具和 RCS 源代码控制系统。
- 《Advanced Programming in the UNIX Environment》[Stevens，1992] 包括了最总要的 Unix 和类 Unix 系统，虽然它预先介绍了 Linux。它有 Linux 应用编程后两章相似的材料：系统调用和共享库。它用了许多例子并解释各种 Unix 版本的区别。
- 《UNIX Network Programming》[Stevens，1990] 完整的介绍了网络编程，包括 Linux 上不实用的网络遗留风格，至少向我们写的这样。读这本书时，从 Berkeley 的 socket

界面（参见 16 章）到维护最大可移植性都很吸引。如果你需要做一些轻微的改动，将你的 Linux 网络程序移植到某些版本的 Unix 上去的话，这本书是很有用的。

到 http://www.awl.com/cseng/books/lad/biblio.html 可进一步找到相关书目的列表。

这本书中所有的源代码都是写书时我们已经测试过的例子。所有的源代码的电子版都可以在 http://www.awl.com/cseng/books/lad/src/ 和 ftp://ftp.awl.com/cseng/books/lad/src/ 下找到。为了说清楚，一些短的源代码段只检验系统工作错误而没有检验所有错误。然而，书中和在我们 Web 和 Ftp 站点的完整程序，我们尽力做到检查所有可能的错误（我们不是完美的）。

本书教你可以使用哪些功能并如何将它们组织在一起；我们鼓励你也学一学怎样使用参考文档，其中一大半包括在你的系统中。

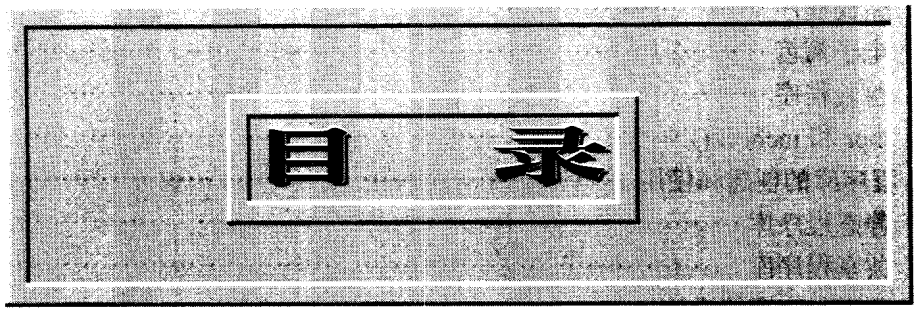
我们欢迎您将建议发到：lad-comments@awl.com 。我们将阅读您的建议，虽然我们不能保证一一回应。

Linux 是一种迅速发展的操作系统，当你阅读这本书时，一些情况（虽然我们不希望）肯定已经改变。我们写这本书时参考 Linux2.0.30 以及在 Red Hat Linux 4.2 上的 C 语言库 5.3.12。我们也已经在 Red Hat Linux 5.0 上 C 语言库 6（glibc 2.0.5）中测试过源代码。

在您的帮助下，我们将在互联网址 http://People.redhat.com/johnson/Lad/errata.html 维护一张勘误和变更表。

我们也感谢每一位技术评论员消耗他们的时间来仔细地思考。他们的建议使这本书更加完善。特别感谢 Linus Torvalds、Alan Cox 和 Ted Ts'o，他们耐心地回答了我们的问题。

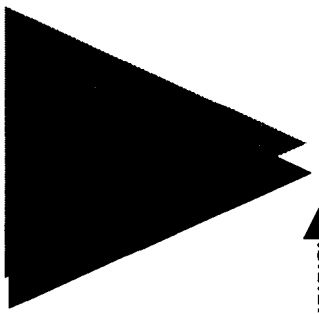
也特别感谢 Kim Johnson 和 Brigid Nogueira。没有他们的耐心这本书就不会写成。



第三部分 系统编程

第四部分 开发库

第一部分



入门

第 1 章　Linux 的发展史

Linux 在人们心目中的定义各不相同，技术上最准确的定义是：

Linux 是一个自由发布的、类 Unix 的操作系统内核。

然而大多数人使用 Linux 是指基于 Linux 内核的整个操作系统。

Linux 是一个自由发布的、类 Unix 的操作系统，它包括系统内核、系统工具、应用程序以及一个完整的开发环境。

由于您将为整个操作系统编写程序，而不是仅仅为内核，所以在本书我们使用的是第二种定义。

Linux（第二种定义）提供了一个很好的可移植程序平台，因为它推荐的接口（我们将在本书中讨论）几乎被所有 Unix 可用版本和大多数克隆版本所支持。学完本书的内容以后，只需少许额外工作，你便可以将程序移植到几乎任一种 Unix 系统或类 Unix 系统上去了。

另一方面，使用 Linux 以后，您可能只愿使用 Linux，并且不再为移植程序而烦恼。

Linux 并不是另一个 Unix。它不仅是一个很好的可移植程序平台，也是一个很好的建立和运行应用程序的平台。Linux 包括以下一些优点：

- 稳定的内核版本是和测试版本完全分离的，而且两者都对大部分公众可用。您可以在一个产品环境中配置稳定的内核系统。Linux 的测试本是经常发布的，如果你愿意的话，它允许你跟随甚至参与操作系统的开发。
- 您可以检查、使用甚至修改整个操作系统的基本源代码。
- 控制 Linux 的是一些个人组织，而不是任一公司。所有的技术决定依赖于技术本身的有利条件，而不是市场销售状况。这意味着 Linux 的发展速度将很快，且能跟上其他相关技术的发展。

1.1　Unix 自由软件简史

如果你想理解什么叫做给 Linux 开发软件，需要上一堂简短的历史课。我们准备的这堂课已经简化并偏重于 Linux 系统的主要元素。想得到更详细、更全面的了解，你可以参考《A Quarter Century of Unix》[Salus,1994] 这本书。

在计算机发展的早期，软件被看作硬件的一个附件。人们想方设法去销售的是硬件，因此很多公司放弃了系统软件。各种进步，如新的算法、新的概念在学生、教授和公司研究者中自由流通。

没过多长时间，公司便认识到了软件是知识产权的价值。他们开始强调软件的版权和限制源代码的发布。软件被看作公共财产，这个变革严格地保护了公司资产，计算机软件发展的文化发生了变化。

世上任一地方的技术革新被商用目的所控制，这是麻省理工学院的 Richard Stallman 所

不希望的。他在马萨诸塞州的剑桥建立自由软件基金会（FSF），FSF 的目的在于鼓励免费发布软件的开发和使用。

“自由（Free）”这一词在文中造成了很多误解，然而 Richard Stallman 所指的“自由”是指自由发布，而不是零消费。他坚信软件及其相关联文件应该和源代码一起销售，对再发布也不应有所限制。为了实现这个理想，在其他人的帮助下，他创建了通用公共许可证（GPL），本书附录 C 中有它的完整副本。

通用公共许可证主要有三点：

1．任何得到通用许可软件的人有权不需额外费用（不包括递送费用）得到其源代码。

2．任何通用许可软件的派生软件必须保留通用公共许可证作为再发布的许可证形式。

3．任何拥有通用许可软件的人有权在不违反通用公共许可证条款的条件下再发布该软件。

在以上条款中值得注意的一点是它没有提到价格（不允许作为额外花费的源除外）。通用许可软件可以以任何价格销售给顾客。但是大多数顾客有权再发布该软件及其源代码。随着 Internet 的出现，这种权利使通用许可证软件在多数情况下保持低价格甚至零价格，在很多情况下，同时允许公司销售通用许可证软件及其服务，例如软件的维护及升级。

通用公共许可证的第二部分引起了很多争议：所有派生软件也必须获得通用公共许可证。尽管诽谤者们把通用公共许可证当成一种病毒，支持者们坚持该条款是通用公共许可证最强有力的部分。它防止公司占有通用公共许可软件，为软件添加特性并将利润饱入私囊。

自由软件基金会发起者们的主要项目是 GNU（GNU is Not Unix 的简称）项目，他们的目标是创立一个自由发布的类 Unix 操作系统。GNU 项目刚开始的时候，高质量的自由发布软件并不多，因此这方案的献身者们乐于为系统开发应用程序和工具更甚于开发操作系统。由于通用公共许可证也是自由软件基金会创建的，GNU 的许多关键组件是按通用许可证发布的，多年来 GNU 项目吸收了许多软件包，例如 X-Windows 系统、TEX 排版系统，还有 Perl 语言，他们在许可证下都是自由发布的。

1.2 Linux 的发展

1991 年，芬兰的赫尔辛基大学的学生 Linus Torvalds 创立一个项目，只是为了在 80386 上自学低层编程。当时，他运行的是 Andrew Tanenbaum 设计的 Minix 操作系统，因此保持他的操作系统与 Minix 的系统调用和文件系统结构兼容，将使工作更为容易。尽管他在 Internet 上发表 Linux 内核第一版时遵循一个相当严格的许可证，但不久他便将许可证转为通用公共许可证了。

通用公共许可证和 Linux 的早期功能的结合说服了其他开发者帮助开发 Linux 内核。后来，发布了从当时占统治地位的 GNU C 库方案中派生出的一个 C 库的实现，它允许建立本地（native）用户应用程序。本地移植 gcc、Emacs 和 bash 也迅速随之而来。在 1992 年的早期，你可以不必做一些紊乱的工作便可以在大多数的 Intel 机器上安装和启动 Linux 0.95 了。

Linux 项目从一开始就和 GNU 项目紧密相连。GNU 项目的源代码基础变成了 Linux 团体建立完整操作系统特别重要的资源。虽然基于 Linux 系统的重要部分起源于包括加州大学

的伯克利分校和 X 联盟可自由发布的 Unix 代码，但是一个功能强大的 Linux 系统的许多重要部分均直接来自于 GNU 项目。

随着 Linux 的成熟，一些个人致力于通过开发软件包，为新用户简化 Linux 系统的安装和使用，这称做发布版（distribution），同时，完全合理的应用组合在一起，构成了一个完整的操作系统。曼彻斯特计算中心（英国曼彻斯特大学）是早期 MCC 发布版本的领导者之一。紧跟着发布的是 SoftLanding 系统（SLS），它就是后来的 SlackWare。SlackWare 是第一个用户无需太多的 Unix 知识就能够安装和维护 Linux 的发布版本。它的流行引起了有关 Linux 安装、使用和维护的计算机图书的出版狂潮。在 Linux 的发布版本世界中，Red Hat Linux 是一个后来者。由于它集中致力于使 Linux 稳定、安全、容易安装和使用，它已经成为最受欢迎的发布版之一。

除了 Linux 的内核之外，一个 Linux 系统的发布版包括开发库、编译器、解析器、Shells、应用程序、实用程序以及配置工具和其他许多的组件。一旦建立一个 Linux 系统，它的发布开发者就会从各个地方收集工具，以创建一个完整集合，包括可实现 Linux 功能的系统所需要的所有软件组件。大多数的发布版也含有使得安装和维护 Linux 系统变得容易的定制组件。

Linux 有很多发布版本。每个版本都有它自己的优点和不足之处；但是，它们都有将 Linux 系统和其他操作系统区分开来的通用内核和开发库。本书的目的是帮助开发人员在任意的 Linux 系统上建立程序。因为所有的 Linux 发布版本都使用相同的代码来提供系统服务和程序的二进制码。而且发布版本之间的源代码的兼容性很高。

对这个兼容性问题作出了贡献的项目是 Linux Filesystem Standard（FSSTND），它定义很多文件应该在哪里保存和解释。总的来说，就是剩下的文件系统应该如何组织起来。这个文献的一个新版本“Filesystem Hierarchy Standard”（FHS）正在起草之中。关于这两个标准的信息都可在 http://www.pathname.com/fhs/中找到。

1.3 Unix 系统的基本家谱

虽然 Linux 系统的主要代码是独立于传统的 Unix 源码基础开发出来的，但是 Linux 提供的接口深受现存 Unix 系统的影响。

在 20 世纪 80 年代的早期，Unix 的开发分成两大阵营，一个在加州大学伯克利分校，另一个在 AT&T 的贝尔实验室。每个机构都开发和维护起源于由贝尔实验室实现的原始 Unix 的 Unix 操作系统。

伯克利版本的 Unix 就是所谓的伯克利软件发布版本（Berkeley Software Distribution, BSD），它在教育界是很流行的。BSD 系统是第一个包括了 TCP/IP 网络的系统，因此取得了很大成功，并促使 Sun 公司建立了第一个基于 BSD 的操作系统 SunOS。

贝尔实验室也在为提高 Unix 系统的性能而努力，但不幸的是，它的做法跟伯克利团体有点不同。贝尔实验室发布的是按照罗马数字标识的子系统。贝尔实验室最后发布的 Unix 版本是 System V（SysV）；现在的 System V Release 4（SVR4）为大多数的商用 Unix 操作系统提供了代码基础。

正是 Unix 的这种分裂发展引起了 Unix 系统在系统调用、系统库和基本命令上的主要

不同。说明这个区别最好的例子之一，就是在每个操作系统提供给应用程序的网络接口上。BSD 系统使用 Socket 实现程序在网络上互相对话。相反，System V 提供的是传输层接口（Transport Layer Interface, TLI），它和 Socket 是完全不兼容的，但它在很多领域上有更大的灵活性。这种发展大大地降低了 Unix 版本之间程序的可移植性能，对第三方给所有的 Unix 版本提供产品的厂商来说，则提高了成本，降低了可行性。

另一个 Unix 系统之间的不兼容例子是 ps 命令，它允许用户查询操作系统的进程信息。在 BSD 系统上，ps aux 给出了所有正在机器上运行的进程完整列表；在 System V 上，该命令是无效的，可以用 ps –ef 命令代替。当然，输出格式跟命令行参数一样是不兼容的。

为了尝试使当时分裂发展导致的 Unix 的各种分歧（就是当时为人们广泛所知的 Unix 战争）标准化，Unix 工业界发起制定了一组定义为 Unix 提供接口的标准。处理编程和系统工具接口的那部分标准就是 POSIX，它由美国电气与电子工程师协会（Institute for Electrical and Electronic Engineers, IEEE）发布。

1.4 Linux 的家谱

“关于标准的最好一点就是可以有许多的选择。”[1] 当 Linux 开发者设计 Linux 的时候，他们有 20 年的历史去检验，更重要的是他们有一个高质量的参考标准。Linux 主要是根据 POSIX 设计的；在放弃 POSIX 的地方，Linux 通常遵循 System V 惯例，但带有一个提供一些 BSD 接口方法的兼容库。值得注意的区别包括：网络互连接口，那是完全的 BSD，甚至不提供 System V TLI 接口的方法；还有一些实用程序，包括网络功能，它们遵循 BSD 模型。

System V 提供 STREAMS 接口，但是 Linux 不实现 STREAMS，因为该接口的性能很差，而它的效果可以用其他方法实现。这个缺点还会影响其他的编程环境；Linux 使用 BSD ptys 取代 System V ptys（唯一的区别在于它们的分配上，它们使用相同的方法），因为 System V ptys 是需要在 STREAMS 的克隆装置上建立起来的。

从编程的角度来看，SVR4 和 Linux 最大的区别是 Linux 并不提供那么多的编程接口副本。甚至程序员专门为 SVR4 系统编写代码时都会更加喜欢用伯克利 Socket 而不使用 TLI；Linux 避开 TLI 的系统开销，执行 Socket。Linux 提供堆栈规则和网络驱动（在内核中），而不是通用的 STREAMS 接口，在保持它用的最多和最有用的功能的同时，去除了它性能上的不足之处。

注 1：Andrew Tanenbaum，Computer Network，Prentice Hall。

第2章 许可证与版权

刚进入自由软件世界的新手经常会对各种与自由软件相关的许可证术语感到迷惑。一些自由软件的热爱者把普通的词汇变成专业术语，然后他们期望人们能够理解每个相关术语的细微差别。

为了编写和发布在像Linux那样的自由平台上的软件，你必须理解许可证和版权。而这些东西总是被那些博学广闻的人，包括自由软件的热爱者，弄得非常复杂。无论你想编写自由软件还是商用软件，你必须要使用许多与许可证术语相关的工具。对版权和许可证的总体理解将会帮助你避免常识性的错误。

在这个好诉讼的时代，要提醒你，我们不是律师。本章反映了我们对所讨论的这些内容的理解，但是并不提供法律上的建议。在你作出任何决定之前，请考虑一下自己和其他人的知识产权问题。你应该进一步学习这个学科，除非你认为没有必要，否则的话，请向律师咨询。

2.1 版权

我们首先讨论比较简单的主题。版权就是对某类知识产权的简单声明。根据最近的国际版权大会，你甚至不需要声明你创作的资料的版权。除非你明确放弃拥有权，其他人只有在严格规定的情况下才允许使用你的知识产权，叫做合理使用；除非你明确给予他们许可权，叫做许可证，否则的话，别人不允许使用你的资料。因此，如果你编写一本书或一个软件，你并不需要写上“Copyright@年份”，以声明拥有版权。但是，如果你不把这个短语写在你的作品上的话，你可能会发现，如果有人侵犯你的版权（通过声明或行为，就好像你并没有拥有版权一样）或许可证时，要在法庭上声明拥有权会更加困难。保护版权协定和执行国际组织伯尔尼版权大会[1]要求参与的国家只在以下情况下执行版权：

自首次出版时起，在作者或其他版权所有人授权下出版的所有副本应有版权符号©，跟随着版权所有人的姓名和首次出版的年份，以下均应以此方式和位置放置，以表示合理的版权声明。

通常用序列号（c）取代一个圆圈中一个小写字母c这个符号，但是法庭还没有支持这样做。当声明你的版权时，除了系列号（c）外，请记住使用Copyright这个词。

版权并不是永久的。所有的知识版权最终都会在公众域（pubic domain）中湮没。比如说，公众最后会拥有版权的所有权，任何人可以使用所有权做任何事。一旦所有权进入了公众域，就不必遵守许可证条款。有一个曲解：如果在公众域作品的基础上创建一部衍生作品，然后你就可以对你的修改拥有版权。因此，虽然很多旧书现在版权都过了期，它们的版权已经湮没在公众域中，但是编者经常在这里或那里做一些改动，或改正原文中的一

注1：http://www.wipo.org/。