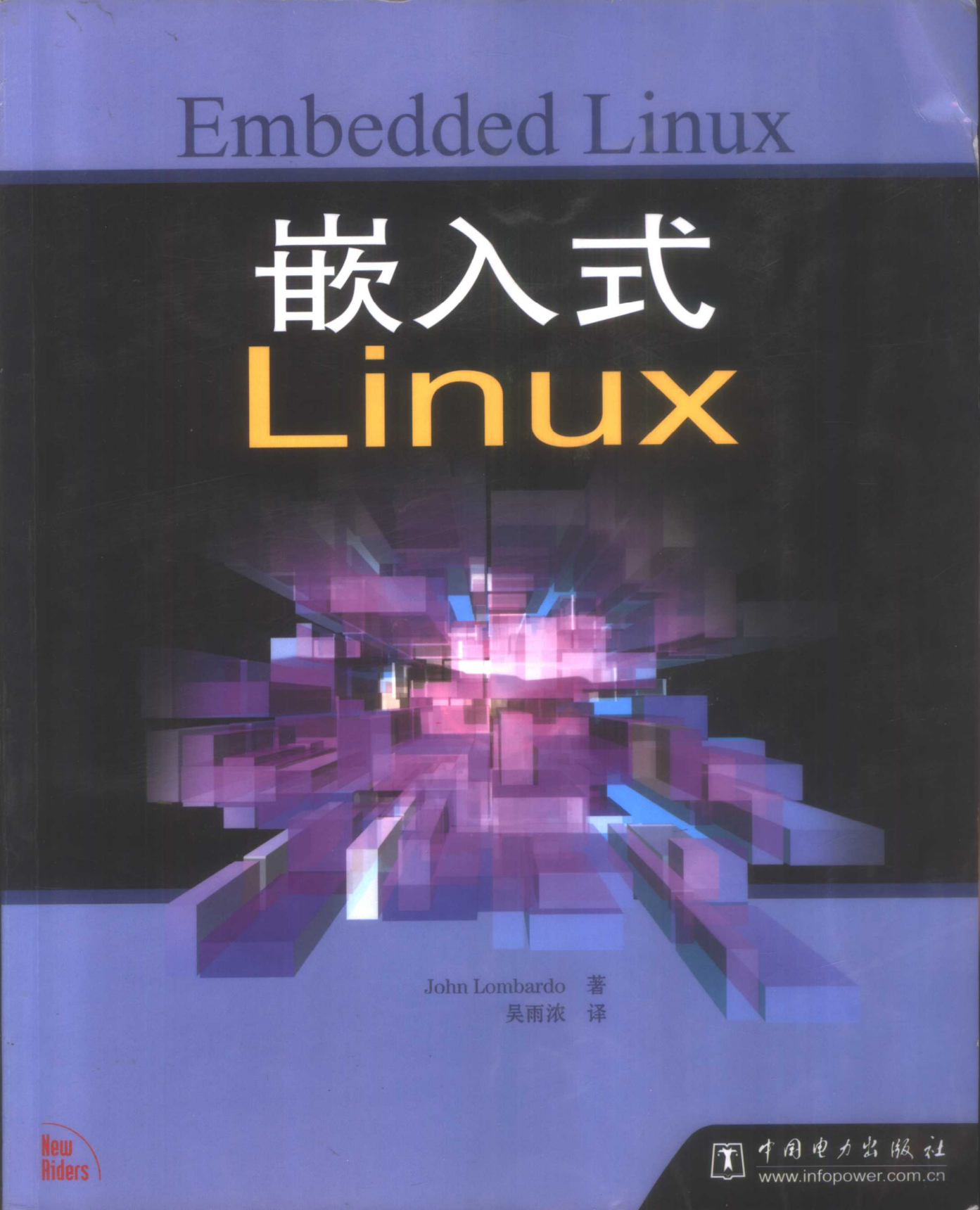
Embedded Linux

嵌入式 Linux



John Lombardo 著
吴雨浓 译

New Riders
中国电力出版社
www.infopower.com.cn

Embedded Linux

嵌入式 Linux

John Lombardo 著
吴雨浓 译

中国电力出版社

Embedded Linux(ISBN 0-7357-0998-X)

北京市版权局著作权合同登记号 图字：01-2002-2013 号

图书在版编目（CIP）数据

嵌入式 Linux /（美）隆巴多（Lombardo,J.）编著；吴雨浓译. —北京：中国电力出版社，2003

ISBN 7-5083-1402-6

Ⅰ.嵌... Ⅱ.①隆...②吴... Ⅲ.Linux 操作系统 Ⅳ.TP316.89

中国版本图书馆 CIP 数据核字（2003）第 004453 号

责任编辑： 常虹

书　　名： 嵌入式Linux

编　　著： （美）Lombardo

翻　　译： 吴雨浓

出版发行： 中国电力出版社

地址：北京市三里河路6号　邮政编码：100044

电话：（010）88515918　　传真：（010）88518169

印　　刷： 汇鑫印务有限公司

开　　本： 787×1092 1/16　　**印　　张：** 13　　**字　　数：** 217千字

书　　号： ISBN 7-5083-1402-6

版　　次： 2003年9月北京第一版

印　　次： 2003年9月第一次印刷

定　　价： 28.00 元

前　言

关于作者

John Lombardo 很早就开始从事嵌入式 Linux 的工作。他的 ShareTheNet 软件运用 Linux 帮助初学者在老式的 X86 计算机上迅速构建高性能路由器。随后，他基于 ARM7 的 NAT 路由器和这本书做了几个嵌入式 Linux 项目，其中包括易于使用的 IPSec 路由器。John 拥有计算机科学学士学位。

关于本书的技术审校

本书的技术审校把他们自己在嵌入式 Linux 开发过程中的实践经验融入了本书。在本书的写作过程中，他们对本书涉及到技术的内容以及本书的组织结构提出了宝贵的意见。他们的反馈保证读者看到了高质量的技术信息。

Erik Andersen

毕业于 Brigham Young 大学，拥有机械工程（mechanical engineering）学士学位，同时辅修数学。另外他也完成了 BYU 大学制造工程（manufacturing engineering）硕士学位的全部课程，但是由于他的女儿 Jessika 的降生，他还没有最终取得学位就开始为一家机器人技术公司工作，主要从事工作单元（workcell）开发、动力学、计算机视觉方面的工作。

Erik 从 1994 年开始使用 Linux，并于 1996 年成为 Debian 的程序员。在他购买了 CD-ROM 驱动器以后发现其实 Linux 内核对于这类设备的支持很少，因此，1996 年~1998 年之间他成为 Linux 内核对于 CD-ROM 子系统驱动支持部分代码的维护者。在这段时间中他定义了内核的 CD-ROM 接口，并在内核中增加了对 IDE-CD 的支持，同时升级了内核 CD-ROM 的接口，移植了很多旧的但是很有用的 CD-ROM 驱动程序到新的接口下，并把这个新的接口命名为“统一 CD-ROM 驱动（Uniform CD-ROM

driver)”。

Erik 是 Lineo 公司雇佣的第一个 Linux 工程师。他曾经是 Embedix Linux 1.0 开发小组的组长，成功地把 Linux（uClinux）的内核移植到了 Atmel AT91 系列 CPU（ARM7TDMI 核心）上，并维护了合作开发网站 http://opensource.lineo.com/ 和 http://cvs.uclinux.org/。同时 Erik 还维护了一些被广泛使用的嵌入式 Linux 软件，例如 BusyBox（一个微型实用工具集）和 uClibc（嵌入式 Linux 的 C 库）等。

两年以来他一直是盐湖城 Linux 用户协会的主席，他跟夫人和两个孩子都住在那里。

Ingo Cyliax

他毕业于 Purdue 大学，拥有计算机和电子工程项目学位，并一直为很多大学和公司提供硬件和软件工程咨询。

他在印第安那大学的回旋加速器实验室（IUCF）工作，他的主要工作是设计实用的电子探测器和数据采集系统，例如 STAR。IUCF 的主要工作方向是癌症护理和辐射对电子原件的影响。现存于 BrookHaven 国家实验室的 STAR 探测器被用于研究“大爆炸”发生后的极短的时间内发生的事件。

在他的空闲时间，他写过关于“Circuit Cellar INK”的文章，其中关于“实时 PC（Realtime PC）”的部分就是由他主笔的，也写过各种关于嵌入式应用的文章，例如机器人控制 FPGA（现场可编程阵列）的遗传算法以及其他感兴趣的领域。同时他还是《Real-Time Programming: A Guide to 32-Bit Embedded Development》（Addison-Wesley, 1999,ISBN 0-201-48540-0）一书的作者之一。

致谢

写一本书需要做大量的工作。如果没有如下朋友的帮助，本书的出版是不可能的，因此本书的出版在很大程度上要归功于他们的辛勤努力。但是请注意，本书的任何错误和疏漏都是本书作者的责任——请给我写电子邮件而不是他们。

Erik Andersen 帮我修正了一些技术问题，其中包括我对于“实时”的错误描述。同时非常感谢 Ingo Cyliax 和 Mark Whitley 提供的反馈和修改意见。

我还要感谢如下朋友：

Robin Drake 为本书的出版做了大量的幕后工作，凭借他对出版业的深刻了解，引导我完成了本书写作的全过程。

John Keefe 为本书做了一步一步的详细检查。

Laurie Petrycki，New Riders 出版公司的编辑，给了我写作这本书的机会。

Bill Shields 提供了 Rick Lehrbaum 的文章“Doctor Dobbs Journal”，并为本书写了有关硬件的章节。

Ann Quinn，本书的采集编辑，他给予了我大量的支持和鼓励。

我还想感谢我的妻子 Dena 和我的孩子 Andrew、Cameron 和 Zachary，在我利用所有的时间间隙写作的一年中一直支持我。

告诉我们你的想法

作为本书的读者，你是本书最重要的评论员。我们重视你的观点和你所做的事情，请告诉我们你认为我们怎样做更好、你认为我们出版哪些领域的图书会更有用，以及把任何你认为需要告诉我们的东西告诉我们。

作为 New Riders 出版社的执行编辑，我们欢迎你与我们联系，你可以通过电子邮件、传真联系我们，或者也可以直接写信给我们，告诉我们你喜欢或者不喜欢本书，并给我们把本书做得更好的一些建议。

但是请你注意，我们不能帮助你解决与本书有关的技术问题，面对大量的邮件，我们很抱歉可能不能一一回复。

在你给我们写邮件的时候请注意要包括本书的名字、作者以及你的名字、电话或者传真号码。我们将仔细地阅读你的邮件并与本书的作者和编辑一起分享你的邮件。

传真：317-581-4663

邮件：Stephanie.Wall@newriders.com

地址：Stephanie Wall

Executive Editor

New Riders Publishing

201 West 103rd Street

Indianapolis, IN 46290 USA

简　介

辛苦了一天，你轻松地回到家，把钥匙扔到电视前面的桌子上，几秒钟以后，用遥控器打开电视，收看你喜欢的 I Love Lucy 马拉松节目。这个可以自动记录电视节目的产品是 TiVo，TiVo 的核心就是 Linux。

我们生活在一个迅速自动化的世界中。我们的桌面上有计算机，但是更多的时间我们还是处在墙壁、家具和衣服的质感中间。它们是那样容易使用，所以我们不会感到无从入手，从它们的物理设计上我们就可以轻易地发现它们的使用方法。其实，其中很多设备的核心就是 Linux 操作系统。

Linux 是一个叫做 Linus Torvalds 的刚毕业的大学生在 1990 年发明的。Linux 现在已经成为一种主要的操作系统，它的影响迅速地从黑客爱好者开始，遍及全球的 IT 部门，直到现在遍及普通大众。实际上，VA Linux 公司的股票在交易第一天就上长了 733%——金融界的新记录！

为什么是 Linux?

为什么 Linux 如此流行？为什么是它驱动嵌入式计算机应用风靡全球？我们可以找出很多的原因，其中最重要的是：

- Linux 是一个开放式的操作系统

Linux 与其他操作系统相比，具有一个怎么强调也不过分的特性，那就是它的开放本质。在操作系统市场，Linux 的主要竞争对手是微软的 Windows 操作系统家族。其他很多基础很好的公司也提供一些操作系统产品在市场上与微软竞争，在很多时候这些产品的技术是很优秀的，但是它们都没有成功地取得预期的市场

份额。因为与 Windows 一样，所有这些产品都具有专利权，它们的源代码也是不公开的。

但是与之相反，Linux 的源代码却是完全公开的，任何人都可以任意修改它们。你可以自己建立一个 Linux 版本，并把它销售给任何人，而不需要支付任何版税。你也可以对 Linux 做一些必要的修改以便其在市场竞争中取胜。惟一的要求是你必须把你对于内核的全部修改也公开给其他人。

“开源”概念实实在在地改变了计算机工程——特别是在价值链的顶端和底端——可以怎样设计和应该怎样设计的思路。如果你正在设计一个大型的多用户的计算机系统，也许是一个大型网站，却没有考虑把 Linux 作为一个适合的解决方案，那将是非常遗憾的。最后也许你并没有选择 Linux，但是你至少应当考虑 Linux。对于价值链的另外一端——那些非常小的计算机系统，例如嵌入式设备也是一样的，也许最后你由于某种原因选择了其他的操作系统，但是你还是应该主要考虑 Linux 的开放源代码的本质和它带给桌面系统的好处。

- Linux 的高程度的知识共享是一种动力

实际上存在着另外一种比较老的，但是从某种角度来说技术上非常先进的开放源代码的操作系统，叫做 FreeBSD。有很多讨论，为什么 Linux 可以在程序员社区获得如此巨大的成功，但是 FreeBSD 却不能获得同样的认同呢？但是无论是什么原因，在过去的几年中 Linux 都获得了广泛的认可和接受。这种“接受”转化为一种知识的共享，这种“知识共享”又转化为一种动力，这种“动力”反馈回来的是更高程度的对于 Linux 的接受。这种良性循环使得 Linux 增长越来越快，同时市场回报也越来越高，这是 Linux 在今天如此流行的主要原因。

什么是嵌入式系统？

好，现在我们已经知道了为什么 Linux 是一种优秀的操作系统的原因了，过一会我们再就这个主题讨论更多的细节，目前还有一件重要的事情，那就是我们需要准确地理解什么是“嵌入式”系统。

用户界面

也许一种简单的区分嵌入式系统与通用计算机系统不同的方法就是用户界面。通

用计算机的用户界面通常包括监视器、键盘、一个鼠标和一些与它接驳的打印设备。而嵌入式也许根本就没有用户界面，或者是有一些很特殊的主界面，例如按钮触摸屏，或者一个大的控制面板。缺少用户界面的嵌入式系统通常只是监听网络或者传感器收集的数据，或者发送和接收命令。你也许可以在一个嵌入式系统上接上一个监视器或者一个键盘，但是这不是操作的正常模式。通常只有在调试或者配置的时候才把它们插上。

那么是不是那些你的 ISP 摆在架子上的，没有监视器和键盘的 PC 也是嵌入式系统呢？恐怕不是——看来我们的定义还需要更精确一点。

有限任务

多数情况下，要确定一个计算机系统适合普通的计算机系统还是嵌入式系统，需要对系统生命周期内的任务做一个考察。嵌入式系统是针对一定的问题或者特定的设备的、基于成本考虑的解决方案。而通用的计算机系统则更像是瑞士军刀，它们在出厂的时候就被预先安装在计算机上，并没有明确的任务。瑞士军刀可以砍、剪、刮、削、钻等等，与此类似，通用系统可以处理工资单、玩 Quake 游戏、上网冲浪等等。

嵌入式系统是具有有限任务的系统，它也许可以做 10 件事情，而且只能做这 10 件事情。例如，安装在你的汽车上的计算机，它可以检测你的燃料状况，收集危险的发动机遥感信号，同时等待为特定命令服务等等，它不停地只是做这些事情——不论它多么强大，你也不能在它上面玩 Quake。

虽然嵌入式系统是有限任务系统，但是并不意味着任务不能增加。例如，Cisco 的路由器就可以由用户升级到最近的软件版本。新的软件不仅可以修复旧的软件中的错误，也可以增加新的功能——这意味着嵌入式系统被扩展了。

另外一个关于嵌入式系统任务扩展的有趣例子是 NASA（美国宇航局）的航海者 1 号和 2 号宇宙飞船的故事（参见 www.jpl.nasa.gov/releases/97/vgrani97. html）。航海者 2 号是这两艘飞船中首先被发射的一艘，发射的时间是 1977 年 8 月 20 日。几个星期以后，也就是 9 月 5 日，航海者 1 号也发射了。开始的时候两艘飞船都只支持两个任务——木星和土星探险。但是人们不能相信，在完成对于木星和土星的探险以后，两艘飞船的状况还非常好，于是 NASA 决定扩展其中航海者 2 号飞船的任务，去探

索天王星和海王星。由于飞船要飞越太阳系，远程控制给飞船提供了比在地球上更强大的能力。

> 任何对于该嵌入式计算机感兴趣的读者都可以访问 NASA 的站点，查看“Computers On Board Unmanned Spacecraft”，网址为：http://www.hq.nasa.gov/office/pao/History/computers/contents.html。

硬件成本与软件弹性之间的比率

我们生活在这样的一个时代，嵌入式计算机的硬件部分，例如 CPU、内存、闪存的价格大跳楼，与之相适应的，对于需要在嵌入式环境下运行的软件的需求则呈现爆炸状态。综合以上这两点，这两个趋势改变了嵌入式工程师的规则。过去工程师们总是在考虑如何利用计算机内存和外存中宝贵的最后一位空间，而现在工程师则把主要的精力放在考察如何提供更多的新的软件满足市场需求上。

这个市场中开放源代码的嵌入式 Linux 意义重大。嵌入式 Linux 的解决方案可以比其他操作系统更快速地被拿到市场上来，因为 Linux 的源代码是公开的，有比其他操作系统更多的人在 Linux 上开发。有一种观念认为那些嵌入式 Linux 的开发人员是最糟糕的黑客，他们不知道怎样去开发程序，这种观念是完全错误而不符合现实的。像 Red Hat 和 VA Linux 这样的公司会花费大量的金钱，请来最好的技术人员，把 Linux 完善到最好。

软件成本

最明显的——但是不一定是最重要的——优势是，Linux 不需要你支付任何许可证费用、版税，或者源代码许可证费用。事实上在你为私人目的使用 Linux 的时候不需要向任何人支付任何费用。但商业化的嵌入式操作系统，像 WindRiver 公司和 QNX 公司的产品，从开发一开始你就需要支付上万美元，而你的产品做好以后，每销售一个产品还需要向他们支付版税。

当然，对于厂商来说，嵌入式 Linux 操作系统从开发成本、维护成本，甚至从很长时期的逐步升级的角度来看，成本都近似于 0。

稳定性

很多人都听说过这样一种说法，就是 Linux 的稳定性非常好，尤其是在与微软公司的 Windows NT 产品比较时。那么有没有办法把这个问题量化呢？有人长时间地计算过 Windows NT 蓝屏的频率和 Linux 出现 Oop 的频率吗？有一个好消息，一个叫做 Uptimes 的项目（www.uptimes.net）做了一些了不起的工作来回答这个问题。他们的调查数据显示，Linux 作为第二稳定的操作系统，仅仅排在 NetBSD 和 FreeBSD 后面。到 2000 年 3 月底，Windows NT 在列表中最长的稳定运行时间排第 277，达到 76 天。如果你运行过 Windows NT 的话，你会认为这已经是一个非常好的成绩了，但是同样的环境下，最好的 Linux 已经稳定运行了 575 天，而 BSD 稳定运行的天数则高达 1411 天，这完全没法比较！

可移植性

Linux 已经被移植到了多种不同的硬件结构上。主流的 Linux 源代码与下面所有的 CPU 结构都兼容：

- DEC Alpha（AXP）
  www.alphalinux.org
- ARM 和 StrongARM 处理器
  www.arm.linux.org.uk
  www.uclinux.org
- Hitachi SuperH
  www.m17n.org/linux-sh
- IA-64
  www.linuxia64.org
- i386
- Motorola 68K 系列
  www.linux-m68k.org
  www.uclinux.org
- MIPS 处理器
  www.oss.sgi.com/mips/

- Motorola PowerPC
  www.linuxppc.org
- S390
  www.ibm.com
- Sparc（32 和 64 位）
  www.ultralinux.org

进入市场时间

在过去的 50 年当中，技术进步的速度越来越快，毫不停歇，特别是在计算机应用领域，有人甚至用“Internet time（因特网时间）”来形容这种现象。

一方面像 Intel 和 Motorola 这样的大公司以极快的速度推出新的计算机芯片集与对手竞争；另外一方面，成百上千的软件与因特网公司制定了新的协议和方法加强计算机与人之间的互动。这使得嵌入式操作系统公司，像 WindRiver 和 QNX 都面临巨大的压力。它们必须第一时间或者稍晚一点就支持新出现的硬件技术，而且它们不得不一直坚持支持这种技术。但是对于嵌入式操作系统来说，真正的问题却应该是如何支持软件。它们不得不在软件方面经常打补丁。最新的软件技术很多都是使用 Linux 开发得到的，因此它们首先在 Linux 上可以应用。而非 Linux 上的技术通常就是建构在微软的 Windows 系统上。因为 Linux 拥有大量的开发团队，因此，Windows 下的技术也被很快地移植到 Linux 上来。

Linux 可以快速进入市场的另外一个原因是你可以使开发主机与目标主机使用相同的操作系统。

开放源代码

除了成本，首先需要考虑的问题就是技术。例如，Linux 非常稳定，其他商业化的操作系统也是如此。让 Linux 脱颖而出的是它开放源代码的特性。实际上在考虑所有的技术问题以前就首先应该将 Linux 作为操作系统的可选方案，除非因为某些特殊原因将之删除。

那么开放源代码是什么意思呢？

开放源代码定义的 1.7 版（www.opensource.com）

免费再发布

如果被发布的软件由不同来源的程序组成，那么许可证不得限制任何当事人或组织销售或赠送作为被发布软件成分之一的开放源码软件。许可证不得从此项销售中索取使用费或其他任何费用。

源代码

程序必须包括源代码，必须允许以源代码方式和编译后的形式发布。如果产品的某个部分没有与源代码一同发布，那么必须提供通行的、不需要支付合理范围之外的任何费用的手段以获得源代码——从网络上免费下载是一种可取的方式。源代码必须是程序员对其进行修改的最佳形式。故意地使源代码变得含混晦涩是不允许的。也不允许给出预处理器或翻译器处理的中间结果。

派生作品

许可证必须允许修改软件和派生软件，并且必须允许它们按照原软件的许可证条款进行发布。

作者的源代码的完整性

只有在许可证允许与源代码一同发布“补丁文件”（该“补丁文件”以在创建时对程序进行修改为目的）时，许可证才能对修改形式的源代码的发布进行限制。许可证必须明确地允许发布由修改后的源代码生成的程序。许可证可以要求派生的作品采用不同的名称或不同的版本号以区别于原来的软件。

不得歧视任何个人和团体

许可证不得歧视任何个人或者由多人组成的团体。

不得歧视任何应用领域

许可证不得限制任何人把程序应用于任何领域。例如，不得规定程序不能应用于商业领域或基因研究领域。

许可证的发布

与程序有关的权利必须适用于该程序的任何使用者，并且程序的使用者也不需要为了使用该程序而获得其他许可证的许可。

许可证不能针对于一个产品

与程序有关的权利不能由该程序是否作为某个软件产品的一部分来决定。如果程序从那个发布中被抽出来，并且按照程序的许可证条款进行使用和发布，那么得到该程序的当事人或组织将获得与得到原程序的使用者相同的权利。

许可证不能影响其他软件

许可证不得向与它的软件一同发布的其他软件提出任何限制。例如，许可证不能坚持要求在同一媒体上发布的其他程序都是开放源代码软件。

开放源代码的好处

- 如果你拥有源代码你可以找出所有的 bug

对于程序员来说，没有什么比调试代码的时候进入死胡同更让人沮丧的了，更糟糕的是，这时你正在调试客户出现的问题。进入死胡同意味着你跟踪你的代码，进入一段你没有源代码的程序，因此你不能跟踪发生了什么，但是问题恰恰出现在这一段。一些时候问题出现在你没有源代码的程序中，但是更多的情况是出现在你自己的代码中——也许是你没有正确地运用那些你没有源代码的程序。但是没有关系——你可以花费几个小时或者几天的时间在你没有源代码的部分，可是还是可能没有发现任何问题。如果你有被调用的函数的源代码，就可以快速地找出问题所在，并花费几分钟把问题解决；或者也许你调用的函数中真的存在 bug——你可以修正这个错误，或者继续

你的工作把它留给这个函数的维护人员去修改。

一些程序员拒绝看他们使用的软件的源代码，他们认为软件中的问题只是他们选择的职业生命中的一种事实。但是这不是真的。那些聪明到可以深入到软件的下一层的开放源代码软件程序员（必要的时候，甚至可以深入到内核），几乎从来都不会在调试中进入死胡同，他们几乎可以调试任何问题。事实上，如果出现了不能被调试的问题，那么这些问题几乎一定是硬件的问题。

- 你可以让别人帮助你修正代码

在 1999 年，我为一个应用软件项目工作，这个软件的用户对于应用有丰富的经验，但是对于 Linux 几乎没有什么经验。因此我们必须制作出完整的软件包，其中包括针对用户拥有的硬件设备的安装程序。

几星期以后我们进入了 beta 测试周期，一个用户发现了一个系统板的问题。我们不仅调试了这个在驱动程序层的问题，还把这个问题以及解决方法提交给了这个驱动程序的维护人。他发现我们的这种解决方法只适合我们特定的用户环境，更好的解决方法可能会牵扯到一系列的问题。最后他们完全修复了这个问题并把新的驱动程序的源代码提交给我们。如果我们没有工作在开放源代码的操作系统上，我们就不可能找出这个问题并自己解决这个问题。如果我们工作在一个非开放源代码的操作系统上，也许我们要花费大量的时间做艰苦的工作，最后问题依然无法解决。

- 你可以了解软件是怎样工作的

在 Linux 下拥有所有驱动程序的源代码写新的驱动程序就变得简单多了。如果代码编写人需要一些特性，他可以参考其他的驱动程序，看这些驱动程序是如何实现这些特性的。实际上写一个新的驱动程序的第一步通常是寻找一个跟你要写的类似的驱动程序，复制它，然后在它的基础上根据你自己的实际需要做一些修改。

- 完善的文档

一些开放源代码软件没有完备的文档。但是另一方面，Linux 却有非常完善的文档。Linux Documnt Project 提供了很多的自由文档以及十几本书。当然，软件包如何工作的最终权威还是源代码本身，而这对于所有的开放源代码产品都是可以完全得到的。

开放源代码的缺点

- 保护智力成果很困难

有很多的市场原因可以让你不能或者不想公开你的智力成果：你想在竞争中保持领先地位，你的智力成果基于一些不全属于你的代码等等。如果你有一项智力成果是你不想跟自己身外的世界分享的，那么你要小心这其中不能包含任何一点其他的开放源代码项目的代码，即使为此你不得不牺牲一些时间而耽搁产品进入市场。如果你使用了开放源代码软件项目中的代码，你的软件按照 GPL 就必须开放源代码，成为一个开放源代码的项目——这对于那些想保护自己智力成果的人来说可不是一件好事情。

对于应用程序代码来说，Linux 下也有很多非源代码开放程序（最典型的就是 Netscape），在内核空间中，有很多只有二进制代码的内核模块的例子（例如本书后面会提到的 DiskOnChip）。在内核空间中维护工作的负担意味着高昂的代价，因为内核的很多选项都会改变模块的接口（例如 SMP 与 UP，大的内存模式与小的内存模式，等等），甚至一个内核版本需要为每一个平台编译几次，更严重的是内核接口改变会导致只有二进制代码的内核模块中的专利代码破坏而不得不经常维修，当出现 bug 的时候，标准内核开发人员可能会拒绝帮助他们使用二进制内核模块而要他们参考模块的专利技术。

- 内核发布速度很快，变化很大

Linux 内核变化非常快，不同的版本之间可能存在巨大的差异。如果你一定要把你的程序运行在最新的版本上，这实在是一种苛求。

- 内核并不总是能在你期待的时间内发布

如果你想基于最新的内核提供的新特性发布你的产品，而又不打算发布 beta 测试版本，那么是非常困难的。

- 很难确定你所做的修改是不是能跟系统很好地协调工作

如果你修改了内核代码，你就很难保证它可以与其他代码协调工作得很好。有的时候你修改的内核工作得非常完美，但是却被内核维护者以不符合 Linux 的开发哲学为理由完全拒绝。（你想把那些代码合并，以使其他的人在内核界面改变的时候可以在

你修改的基础上继续维护那些代码。）

- Linux 并不总是保证向上的兼容性

跟其他的操作系统不同，Linux 没有很多跟版本相关的代码。这使得阅读源代码很容易，因为代码中没有很多#ifdef 和其他为了保证兼容而做的代码。但是不幸的是，这意味着一些原来在 Linux 上可以正常运行的程序可能在新的内核上却无法运行。这就使得驱动程序的维护者必须保证驱动程序同步更新，所以你不得不花费一些时间来修正一些由于界面被修改或者服务被删除而造成的、以前没有的 bug。

GNU 对于你有什么影响

如果你准备在你的项目中使用 Linux，那么你必须了解的两个软件许可证是 GNU GPL（General Public License）和 GNU LGPL（Lesser General Public License）。Linux 操作系统的源代码是在 GPL 的保护下，而 GNU C 库是在 LGPL 的保护下。参考附录 A 和 B 的许可证全文。

GPL 与 LGPL 之间有许多类似

因为 GPL 和 LGPL 都是由自由软件基金会（Free Software Foundation）建立的，所以它们之间非常相似，它们都具备如下的重要特性：

- copyleft

术语 copyleft 源于 GNU 项目中的俗语，是英文 copyright 的双关语。它指的是开放源代码许可证中如下条款：任何个人或者公司都不能把 GPL 保护下的软件源代码拷贝、修改，或者修改其许可证，把他们所做的修改保护起来——即使他们的修改可能使得软件的性能比原始版本更好。一旦一段代码被 GPL 或 LGPL 保护，那么代码本身和所有对它所做的修改都必须永远在对它感兴趣的人和组织面前公开。如果第三方修改并发布这段代码，那么必须把修改的代码和原始代码都向接受代码的人公开。

- 有效性

如果你对 GPL 保护下的源代码做了修改，那么最好的方式就是把你所做的修改做成一个补丁包发送给这段代码的维护者，以便你的修改在将来可以作为标准