HZ BOOKS

The Linux kernel Primer

A Top-Down Approach for x86 and PowerPC Architectures

Linux 内核编程

(美) Claudia Salzberg Rodriguez
Gordon Fischer
Steven Smolski
著

陈莉君 贺炎 刘霞林 译

机械工业出版社
China Machine Press

The Linux kernel Primer

A Top-Down Approach for x86 and PowerPC Architectures

Linux 内核编程

(美) Claudia Salzberg Rodriguez
Gordon Fischer
Steven Smolski 著

陈莉君 贺炎 刘霞林 译

机械工业出版社
China Machine Press

本书以 Linux 操作系统为基础，详细介绍了 Linux 内核子系统，并用大量内核源代码和示例程序进行演示，对深入了解 Linux 内核具有指导意义。本书内容主要包括：Linux 基本知识、内核探索工具集、程序执行的基本模型、内存管理、输入/输出、文件系统、调度与内核同步、内核引导、构建 Linux 内核，以及向内核添加代码等。简述一些应用工具和使用程序，从而可以获取理解内核内幕所需的信息。每章末都给出小结和一些练习，涉及内核运行的操作及工作原理。

本书适合不同级别的系统程序员、Linux 爱好者以及应用程序开发人员学习和参考。

本书版权登记号：图字：01-2006-1946

图书在版编目(CIP)数据

Linux 内核编程/(美)罗瑞吉(Rodriguez, C.S.)著；陈莉君等译. －北京：机械工业出版社，2006.7

书名原文：The Linux Kernel Primer: A Top-Down Approach for x86 and PowerPC Architectures

ISBN 7-111-19217-6

Ⅰ.L…　Ⅱ.①罗…　②陈…　Ⅲ.Linux 操作系统　Ⅳ.TP316.89

中国版本图书馆 CIP 数据核字(2006)第 053916 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）

责任编辑：郭冬艳　秦燕梅

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2006 年 7 月第 1 版第 1 次印刷

186mm×240mm · 25 印张

定价：49.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：(010) 68326294

译 者 序

追本溯源，从本书开始！

Linux 最为人称道的莫过于它的自由精神，所有源代码唾手可得。打开 Linux 内核源代码，我们可以看到熟悉的 C 语言函数和一些陌生的汇编代码。但是，Linux 内核入门很不容易，它之所以难学，在于庞大的规模和涉及的层面。规模一大，就不易现出本来面目，浑然一体，自然不容易找到着手之处；层面一多，就会让人眼花缭乱，盘根错节，怎能让人提纲挈领？

就我们的经验，内核初学者(不是编程初学者)可以从这本书着手。本书三位作者有多年的行业经验。Claudia Salzberg Rodriguez 就职于 IBM Linux 技术中心，进行内核及相关编程工具的开发。Gordon Fischer 为很多设备开发了 Linux 和 UNIX 设备驱动程序。Steve Smolski 在半导体行业已经有 26 年，进行过各种驱动程序和嵌入式系统的开发。他们合作奉献给大家的这本内核入门，是对 Linux 内核编程的有效指导。作者独特的由表及里学习途径使得内核编程更易于理解，从用户空间到内核，把内核内在的实现原理与用户级编程的基本原则相联系，系统地追踪了实现功能。这种途径有助于扩大你所了解的 Linux 知识，加深对内核组成及工作机理的理解深度。

在本书的翻译过程中，更是感到作者软硬件知识的全面、对内容独到的组织方式和娴熟的开发经验。在我们熟知的 x86 平台之外，作者对 PowerPC 的深入讲解，不仅让基于 PowerPC 平台的开发者找到了知音，更为 x86 的开发者打开了一扇新窗户。

为了让本书尽快与读者见面，本书翻译组陈莉君、贺炎、刘霞林、康华和石莉放弃了春节的休息时间，夜以继日。书中不妥之处和错误难免，希读者谅解。

在这消化这本书的基础上，如果你侧重于内核的全面了解，可以进一步研究《Understand Linux Kernel》和源代码本身；如果你侧重于了解内核设计思想，则阅读《Linux Kernel Development》；如果你侧重于实际编程，可以研读《Linux Device Driver》，直接开始动手工作；如果你想有一个轻松的内核学习和实践环节，请访问我们的网站 www.kerneltravel.net。

译 者

2006 年 3 月

序

“有龙在此”，中世纪地图绘制者碰到未知和危险的地方就如此标记，可能你首次敲入如下命令也有这样的感觉：

```
cd /usr/src/linux ; ls
```

你可能也发出“从何开始?”的感叹。“我到底想知道什么？这是怎样放在一起的？本质上又如何工作?”

现代功能俱全的操作系统庞大而复杂。子系统为数不少，它们之间的交互更是错综复杂而且微妙。不错，你的确拥有 Linux 内核源代码(稍后还会详述)，但是，从何处开始，着眼于什么，该以怎样的顺序，远非易事。

本书的编写目的正在于此。一步一步，你会了解到内核的各个部分，它们如何工作，互相之间怎样关联。本书的作者熟知内核，这些知识贯穿于本书的始终，你和内核之间至少会成为好朋友，乃至产生深厚的情意。

Linux 内核是“自由的”软件。Richard Stallman 对自由软件给出了定义[⊖]，所谓自由(freedom)就是让软件是自由的(Free，第一个字母 F 大写)。有两层含义，一方面，运行软件是自由的，这是最基本的自由。另一方面，探究程序如何编写也是自由的。这种自由往往被忽略，实际上，这才是最重要的，因为学习如何做事的最好方式之一就是观察别人如何做事。在软件世界中，那就意味着阅读别人的程序，并了解到他们在哪些地方做得较好，哪些地方做得较差。至少在我看来，在现代计算领域里，GNU/Linux 之所以能变成一股强大的力量，其最根本原因之一就是 GPL 的自由。这些自由，在你使用 GNU/Linux 的每时每刻都会感到其益处，偶尔停下来，试想一下是不是这回事。

有了本书，我们充分利用自由之二，让你有机会深入研究 Linux 内核源代码。你会看到，有些事的确做得不错；同时，你也会看到，有些事也并不尽人意。但是，因为自由之二，你会看到全貌，更重要的是，你会从中学到很多知识。

况且，这也使我走入了 Prentice Hall 开源软件开发系列丛书，本书是首批成员之一。开发这一系列丛书的念头源于阅读程序是学习的最好方式之一。如今，这个世界幸运地享有丰富而自由开放的源代码软件，这些源代码正期待着(或许热切渴望着)被阅读、理解，甚至赞许。这一系列丛书会成为你软件开发学习曲线的领路人，也可以说，通过尽可能多地展示真实的代码，有助于你学到货真价实的东西。

我真诚地希望你会欣赏本书，并从中学到东西。我也期望你会从中得到灵感，从而在自由软件和开源世界开创你自己的事业，参与进来，那无疑是最令人愉快的方式了。

玩得开心！

Arnold Robbins

系列丛书编辑

⊖ http://www.gnu.org/philosophy/free-sw.html。

作者简介

Claudia Salzberg Rodriguez 就职于 IBM Linux 技术中心，负责内核及相关编程工具的开发。在担任 Linux 系统程序员的 5 年多时间里，她一直针对 Intel 和 PPC 平台，进行从嵌入式到高性能系统的 Linux 开发。

Gordon Fischer 为很多低层设备开发过 Linux 和 UNIX 设备驱动程序，并针对 Intel 和 PPC 平台，在各种各样的企业级设置中使用 Linux 内核。

Steve Smolski 就职于半导体行业已经有 26 年。他工作过的领域包括制造，测试，内存、处理器以及 ASICS 的研发，他为 Linux、AIX 以及 Windows 都写过应用程序和驱动程序，还进行过嵌入式操作系统的开发。

前　言

无论是一般的技术还是专业性很强的计算机，对于试图了解它们的人们来说都同样具有不可思议的魔力。技术的发展推动着对已确立的框架和曾经模糊的陈旧概念的重新评估。Linux操作系统已经对促进工业变革和商业营销方式做出了巨大贡献。GNU公共许可证的采用，以及与GNU软件的互动，使围绕开源的各种争论有了共同的标准。开源操作系统如此强大，Linux无疑是一个极其成功的典范，它以无法想像的魔力吸引着世界各地的程序员。

对于数量庞大的计算机用户来说，越来越多的人使用了Linux。有了各种各样的发布版，社团的支持，以及工业后盾，Linux的应用也找到了安全的港湾，它的身影出现在大学、工业应用以及数以千万计的家庭用户中。

使用大潮促进了技术支持和新功能需求的日益增长。这样一来，愈来愈多的程序员发现自己对Linux内核的内幕感兴趣，因为大量现有的(还有快速增长的)军工企业需要支持不同的体系结构和种类繁多的新设备。

Linux操作系统的繁荣和发展归功于把内核移植到了Power体系结构，它覆盖了从高端的服务器到低端的嵌入式系统。随着各公司倾向于购买基于Power PC的系统来运行Linux，想知道Linux在该体系结构上运行机理的愿望就显得日益强烈。

适合的读者

本书可供不同级别的系统程序员、Linux爱好者以及应用程序开发者阅读，这些开发者渴望更好地理解自己的程序到底是如何工作的。只要有C知识，熟悉基本的Linux使用基础，还想知道Linux是如何工作的，那么你就会发现这本书提供了进行这种理解基本而必要的概念，可以说，本书是理解Linux内核如何工作的初级读本。

不管你是编写过在Linux上运行的小程序，还是已开发过系统但正在寻求对某个子系统特性的理解，本书所编写的内容都是你所期待的。

内容组织

本书分为三部分，每部分都提供必要的知识，让读者能顺畅地钻研Linux内幕。

第一部分提供必要的工具，并理解对Linux内幕所进行的探索。

第1章“概述”，叙述Linux和UNIX的历史，罗列很多发布版，并从用户空间的观点简述各种内核子系统。

第2章“内核探索工具集”，描述Linux内核中常用的数据结构和语言的用法，介绍x86和PowerPC体系结构的汇编语言，并简述一些工具和实用程序，从而可以获取理解内核内幕所需的信息。

第二部分介绍了在每个内核子系统中所涉及的基本概念，并分析了执行子系统功能的必要代码。

第 3 章“进程：程序执行的基本模型”，涵盖了进程模型的实现。解释了为何引入进程，并讨论了用户空间到内核空间的控制流，也讨论了内核空间到用户空间的控制流。我们还讨论了进程在内核中是如何实现的，并描述了所有与进程执行相关的数据结构。本章还涵盖了中断和异常，描述了这些硬件机制在每种体系结构中是如何发生的，它们与 Linux 内核又是如何交互的。

第 4 章“内存管理”，描述了 Linux 内核如何追踪和管理用户空间进程的可用内存和内核的可用内存。本章描述了内核对内存分类的方式，以及如何决定分配和释放内存，也详细描述了缺页机制以及它是怎样在硬件上执行的。

第 5 章“输入/输出”，描述了处理器是如何与其他设备进行交互的，内核又是如何响应和控制这些交互的。本章还涵盖了各种设备及其在内核中的实现。

第 6 章“文件系统”，概述文件和目录如何在内核中实现。本章引入了虚拟文件系统，它是用于支持多文件系统的抽象层。本章还跟踪了文件相关操作的执行，如打开和关闭文件。

第 7 章“调度和内核同步”，描述调度程序的操作，调度程序让多个进程运行起来就像只有一个进程在系统中运行一样。本章详细描述了内核如何选择一个任务运行，进程切换时如何与硬件进行交互。本章还叙述了什么是内核抢占，它又是怎样执行的。最后，描述了系统时钟的工作原理，内核怎样使用它计时。

第 8 章“内核引导”，描述电源开和关时都发生些什么。本章对各种处理器装入内核的方式进行了跟踪，包括对 BIOS、Open Firmware 和 bootloaders 的描述。然后，考察了内核启动和初始化时的线性顺序，涵盖了前面章节中讨论的所有子系统。

第三部分，描述如何构建内核并与内核进行交互的多种途径。

第 9 章“构建 Linux 内核”，涵盖了编译内核所必需的工具和执行的目标文件的格式。还详细描述了内核源代码编译(Kernel Source Build)系统是如何操作的，怎样把配置选项加到内核编译系统中。

第 10 章“向内核添加代码”，描述了/dev/random 操作，这在所有的 Linux 系统中都可以看到。就像对熟悉的设备进行描述一样，本章还从更实战的观点触及了曾经描述过的概念。最后，还描述了如何给自己的设备编写代码。

探索方法

本书向读者介绍了理解内核的必要概念。我们遵循自顶向下的方式来组织内容，具体体现在以下两个方面：

首先，我们把内核的工作和用户空间所执行的操作关联起来，因为读者对后者更熟悉，并渴望以这样的方式理解内核的工作原理。在可能的情况下，我们会从用户空间的例子说起，并跟踪代码的执行到内核。但有时，这种跟踪方式并不总是可行，因为子系统的数据类型和底层结构必须在解释其工作原理之前引入。在这些情况下，我们把对内核子系统的解释和它与用户

空间程序如何联系的具体例子结合起来。有双重意图：其一，当内核一方面与用户空间打交道，另一方面与硬件打交道时突出了在内核看到的层面；其二，通过跟踪代码和事件发生的顺序来解释子系统的工作原理。我们相信，这将有助于读者把内核的工作原理与自己所知道的知识相关联，也有利于让读者看到，一个特定的功能是怎样与操作系统的其他部分相联系的。

其次，我们采用自顶向下的方法，考察针对子系统操作的主要数据结构，并观察其怎样与系统管理的执行行为相联系。我们尽力描述针对子系统操作的结构，并像追踪子系统的操作一样持续关注这些数据结构。

目　录

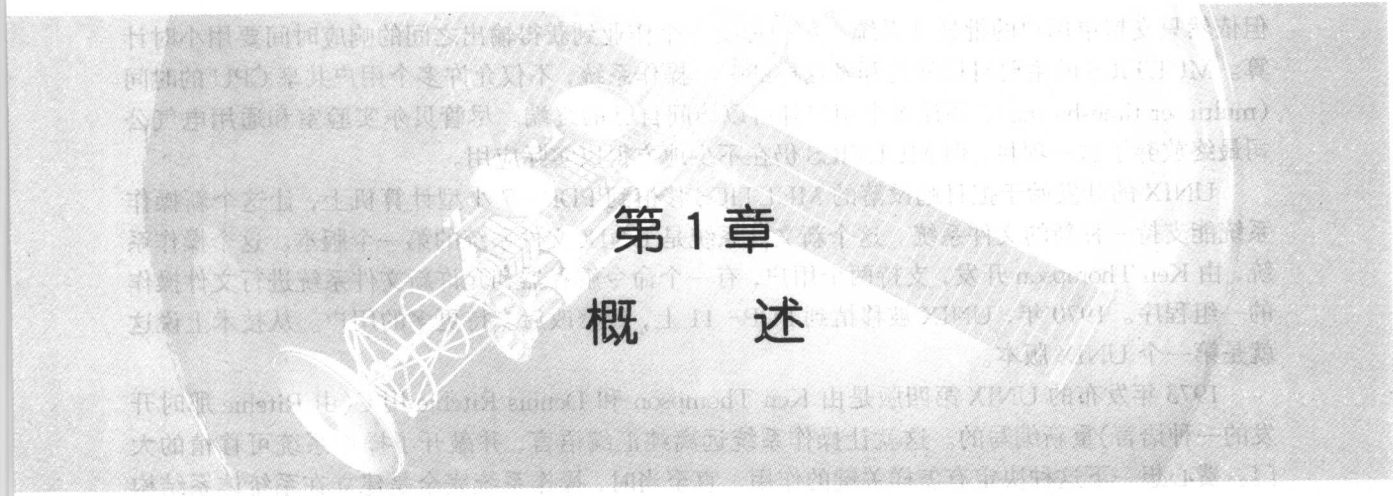
译者序
序
作者简介
前言

第 1 章　概述
1.1　UNIX 发展史 …… 1
1.2　标准和通用接口 …… 2
1.3　自由软件和开放源码 …… 3
1.4　Linux 发布版的快速浏览 …… 3
1.4.1　Debian …… 4
1.4.2　Red Hat/Fedora …… 4
1.4.3　Mandriva …… 4
1.4.4　SUSE …… 4
1.4.5　Gentoo …… 4
1.4.6　Yellow Dog …… 4
1.4.7　其他发布版 …… 5
1.5　内核版本信息 …… 5
1.6　基于 Power 的 Linux …… 5
1.7　操作系统的概念 …… 6
1.8　内核组织 …… 7
1.9　Linux 内核概述 …… 7
1.9.1　用户接口 …… 7
1.9.2　用户身份鉴别 …… 8
1.9.3　文件和文件系统 …… 8
1.9.4　进程 …… 12
1.9.5　系统调用 …… 15
1.9.6　Linux 调度程序 …… 15
1.9.7　Linux 设备驱动程序 …… 16
1.10　可移植性和体系结构相关性 …… 16
小结 …… 17
习题 …… 17
第 2 章　内核探索工具集 …… 18
2.1　内核中常见的数据类型 …… 18
2.1.1　链表 …… 18
2.1.2　查找 …… 21
2.1.3　树 …… 21
2.2　汇编 …… 23
2.2.1　PowerPC …… 24
2.2.2　x86 …… 26
2.3　汇编语言示例 …… 28
2.3.1　x86 中的汇编示例 …… 29
2.3.2　PowerPC 中的汇编示例 …… 31
2.4　内联汇编 …… 33
2.4.1　输出操作数 …… 33
2.4.2　输入操作数 …… 33
2.4.3　修改过的寄存器(或者已修改元素列表) …… 33
2.4.4　参数的编号方式 …… 34
2.4.5　约束条件 …… 34
2.4.6　asm …… 34
2.4.7　_volatile_ …… 34
2.5　特殊的 C 语言用法 …… 37
2.5.1　asmlinkage …… 37
2.5.2　UL …… 38
2.5.3　inline …… 38
2.5.4　const 和 volatile …… 38
2.6　内核探测工具一览 …… 39
2.6.1　objdump/readelf …… 39
2.6.2　hexdump …… 40
2.6.3　nm …… 41
2.6.4　objcopy …… 41
2.6.5　ar …… 41
2.7　内核发言:倾听来自内核的消息 …… 41
2.7.1　printk() …… 41

2.7.2 dmesg …… 41
2.7.3 /var/log/messages …… 42
2.8 其他 …… 42
2.8.1 _ init …… 42
2.8.2 likely()和 unlikely() …… 42
2.8.3 IS _ ERR 和 PTR _ ERR …… 43
2.8.4 通告程序链 …… 44
小结 …… 44
项目：Hellomode …… 44
习题 …… 47
第 3 章 进程：程序执行的基本模型 …… 48
3.1 引入程序 …… 49
3.2 进程描述符 …… 51
3.2.1 与进程属性相关的域 …… 53
3.2.2 与调度相关的域 …… 54
3.2.3 涉及进程间相互关系的域 …… 56
3.2.4 进程信任度相关的域 …… 58
3.2.5 进程权能相关的域 …… 59
3.2.6 进程限制相关的域 …… 60
3.2.7 文件系统和地址空间相关的域 … 61
3.3 进程的创建：fork()、vfork 和 clone()系统调用 …… 62
3.3.1 fork() 函数 …… 64
3.3.2 vfork()函数 …… 64
3.3.3 clone() 函数 …… 65
3.3.4 do _ fork() 函数 …… 66
3.4 进程生命周期 …… 68
3.4.1 进程的状态 …… 68
3.4.2 进程状态转换 …… 69
3.5 进程的终止 …… 72
3.5.1 sys _ exit() 函数 …… 73
3.5.2 do _ exit() 函数 …… 73
3.5.3 父进程通知和 sys _ wait4() …… 75
3.6 了解进程的动态：调度程序的基本构架 …… 77
3.6.1 基本结构 …… 78
3.6.2 从等待中醒来或者激活 …… 79
3.7 等待队列 …… 83
3.7.1 添加到等待队列 …… 85
3.7.2 等待事件 …… 86
3.7.3 唤醒进程 …… 88
3.8 异步执行流程 …… 90
3.8.1 异常 …… 90
3.8.2 中断 …… 92
小结 …… 109
项目：current 系统变量 …… 110
习题 …… 112
第 4 章 内存管理 …… 113
4.1 页 …… 115
4.1.1 标志 …… 116
4.2 内存区 …… 117
4.2.1 内存区描述符 …… 117
4.2.2 内存区操作辅助函数组 …… 119
4.3 页面 …… 120
4.3.1 请求页面函数族 …… 120
4.3.2 释放页面的函数族 …… 121
4.3.3 伙伴系统 …… 122
4.4 Slab 分配器 …… 126
4.4.1 缓存描述符 …… 127
4.4.2 通用目的缓存描述符 …… 131
4.4.3 slab 描述符 …… 131
4.5 slab 分配器的生命周期 …… 133
4.5.1 slab 分配器有关的全局变量 …… 133
4.5.2 创建缓存 …… 134
4.5.3 slab 创建与 cache _ grow() …… 139
4.5.4 Slab 的销毁：退还内存与 kmem _ cache _ destroy() …… 141
4.6 内存请求路径 …… 142
4.6.1 kmalloc() …… 142
4.6.2 kmem _ cache _ alloc() …… 143
4.7 进程内存结构 …… 144
4.7.1 mm _ struct …… 144
4.7.2 vm _ area _ struct …… 146
4.8 进程映像分布于线性地址空间 …… 147
4.9 页表 …… 150
4.10 缺页 …… 150

4.10.1 x86 缺页异常 …… 151
4.10.2 缺页处理程序 …… 151
4.10.3 PowerPC 缺页异常 …… 158
小结 …… 158
项目：进程内存映射 …… 159
习题 …… 160
第 5 章 输入/输出 …… 161
5.1 硬件如何实现总线、桥、端口和接口 …… 161
5.2 设备 …… 165
5.2.1 块设备概述 …… 165
5.2.2 请求队列和 I/O 调度 …… 166
5.2.3 示例："通用"块设备驱动程序 … 174
5.2.4 设备操作 …… 176
5.2.5 字符设备概述 …… 177
5.2.6 网络设备 …… 177
5.2.7 时钟设备 …… 177
5.2.8 终端设备 …… 178
5.2.9 直接存储器存取 …… 178
小结 …… 178
项目：创建并口驱动程序 …… 178
习题 …… 186
第 6 章 文件系统 …… 187
6.1 文件系统的一般概念 …… 187
6.1.1 文件和文件名 …… 187
6.1.2 文件类型 …… 188
6.1.3 附加文件属性 …… 188
6.1.4 目录和路径名 …… 189
6.1.5 文件的操作 …… 189
6.1.6 文件描述符 …… 189
6.1.7 磁盘块，磁盘分区及其实现 …… 190
6.1.8 性能 …… 191
6.2 Linux 的虚拟文件系统 …… 191
6.2.1 VFS 的数据结构 …… 193
6.2.2 全局链表和局部链表的引用 …… 203
6.3 与 VFS 相关的结构 …… 204
6.3.1 fs_struct 结构 …… 205
6.3.2 files_struct 结构 …… 205
6.4 页缓冲 …… 208
6.4.1 address_space 结构 …… 209
6.4.2 buffer_head 结构 …… 210
6.5 VFS 的系统调用和文件系统层 …… 212
6.5.1 open() …… 213
6.5.2 close() …… 217
6.5.3 read() …… 220
6.5.4 write() …… 235
小结 …… 236
习题 …… 237
第 7 章 调度和内核同步 …… 238
7.1 Linux 调度程序 …… 239
7.1.1 选择下一个进程 …… 239
7.1.2 上下文切换 …… 244
7.1.3 让出 CPU …… 251
7.2 抢占 …… 259
7.2.1 显式内核抢占 …… 259
7.2.2 隐式用户抢占 …… 259
7.2.3 隐式内核抢占 …… 260
7.3 自旋锁和信号量 …… 262
7.4 系统时钟：关于时间和定时器 …… 264
7.4.1 实时时钟：现在几点了？ …… 264
7.4.2 读取 PPC 实时时钟 …… 266
7.4.3 读取 x86 的实时时钟 …… 268
小结 …… 269
习题 …… 270
第 8 章 内核引导 …… 271
8.1 BIOS 和 Open Firmware …… 272
8.2 引导装入程序(Boot Loaders) …… 272
8.2.1 GRUB …… 273
8.2.2 LILO …… 275
8.2.3 PowerPC 和 Yaboot …… 276
8.3 体系结构相关的内存初始化 …… 277
8.3.1 PowerPC 的硬件内存管理 …… 277
8.3.2 基于 Intel x86 体系结构的硬件内存管理 …… 286
8.3.3 PowerPC 和 x86 的代码汇集 …… 294
8.4 原始的 RAM 盘 …… 294

8.5 开始：start_kernel() …… 295
8.5.1 调用 lock_kernel() …… 296
8.5.2 调用 page_address_init() …… 298
8.5.3 调用 printk(linux_banner) …… 300
8.5.4 调用 setup_arch …… 300
8.5.5 调用 setup_per_cpu_areas() … 303
8.5.6 调用 smp_prepare_boot_cpu() …… 304
8.5.7 调用 sched_init() …… 305
8.5.8 调用 build_all_zonelists() …… 307
8.5.9 调用 page_alloc_init …… 307
8.5.10 调用 parse_args() …… 308
8.5.11 调用 trap_init() …… 310
8.5.12 调用 rcu_init() …… 310
8.5.13 调用 init_IRQ() …… 311
8.5.14 调用 softirq_init() …… 312
8.5.15 调用 time_init() …… 312
8.5.16 调用 console_init() …… 313
8.5.17 调用 profile_init() …… 314
8.5.18 调用 local_irq_enable() …… 314
8.5.19 配置 initrd …… 315
8.5.20 调用 mem_init() …… 315
8.5.21 调用 late_time_init() …… 320
8.5.22 调用 calibrate_delay() …… 320
8.5.23 调用 pgtable_cache_init() …… 321
8.5.24 调用 buffer_init() …… 322
8.5.25 调用 security_scaffolding_startup() …… 323
8.5.26 调用 vfs_caches_init() …… 323
8.5.27 调用 radix_tree_init() …… 329
8.5.28 调用 signal_init() …… 330
8.5.29 调用 page_writeback_init() … 330
8.5.30 调用 proc_root_init() …… 332
8.5.31 调用 init_idle() …… 334
8.5.32 调用 rest_init() …… 334
8.6 init 线程(或进程 1) …… 335
小结 …… 339
习题 …… 339
第 9 章 构建 Linux 内核 …… 340
9.1 工具链 …… 340
9.1.1 编译程序 …… 341
9.1.2 跨编译程序 …… 341
9.1.3 链接程序 …… 342
9.1.4 ELF 二进制目标文件 …… 342
9.2 编译内核源代码 …… 346
9.2.1 解释源代码 …… 346
9.2.2 编译内核映像 …… 349
小结 …… 355
习题 …… 355
第 10 章 向内核添加代码 …… 356
10.1 浏览源代码 …… 356
10.1.1 熟悉文件系统 …… 356
10.1.2 Filps 和 Fops …… 357
10.1.3 用户空间和内核空间 …… 359
10.1.4 等待队列 …… 360
10.1.5 工作队列和中断 …… 363
10.1.6 系统调用 …… 365
10.1.7 其他类型的驱动程序 …… 365
10.1.8 设备模型和 sysfs 系统文件 …… 368
10.2 编写源代码 …… 370
10.2.1 设备基础 …… 370
10.2.2 符号输出 …… 372
10.2.3 IOCTL …… 373
10.2.4 轮询与中断 …… 375
10.2.5 工作队列和 Tasklets …… 379
10.2.6 增加系统调用的代码 …… 380
10.3 编译和调试 …… 382
小结 …… 383
习题 …… 384
参考文献 …… 385



第1章 概 述

1991年Linux操作系统诞生于当时还是学生的Linus Torvalds之手。与现在所取得的成就相比，最初的Linux显得微不足道。Linux刚开发时运行在具有AT硬盘、x86体系结构的计算机上。第一个发布版支持bash shell和gcc编译器。那时还不关心其可移植性，也没有关注其在学术和工业界是否能够广泛应用，更没有商业计划或远景宣言。但是，它就这么一天天自由地发展起来。

从早期的beta版开始，在Linus的指导和维护下，Linux变成一个合作项目。这填补了一项空白，使黑客们可以使用运行在x86体系结构上的免费操作系统了。这些黑客们根据自己特定的需要对Linux提供支持，开始贡献源代码了。

通常说，Linux是UNIX的一种类型。从技术上说，Linux是UNIX的一个克隆，因为它实现了POSIX UNIX规范P1003.0。UNIX从1969年诞生以来就统治着非Intel工作站舞台，被公认为是一个强大而又雅致的操作系统。UNIX移交给高性能工作站后，只用在研究、学术以及开发机构里。Linux把UNIX系统的能力带入到了Intel个人计算机和其用户家里。如今，Linux在工业和学术领域中得到了广泛的应用，支持众多的体系结构，如PowerPC。

本章简要介绍了围绕Linux的概念，引领你概览内核的组成和特点，并介绍了一些引人入胜的Linux特性。为了理解Linux内核的概念，你需要对其预期目标有一个基本理解。

1.1 UNIX发展史

我们曾提到，Linux是UNIX的一种类型。尽管Linux并不是直接从现有的UNIX衍生而来的，但事实上，它实现了通用UNIX标准，这使得UNIX的历史与我们的讨论密切相关。

MULTICS(MULTiplexed Information and Computing Service)被认为是UNIX操作系统的鼻祖，它是麻省理工学院、贝尔实验室以及通用电气公司(GEC，那时该公司从事计算机制造业)联合起来开发的操作系统。MULTICS的开发源于让机器支持众多的分时用户。这一联合开发的时间是1965年，尽管当时该操作系统支持多道程序设计(multiprogramming，作业之间分时)，

但依然只支持单用户的批处理系统。用户提交一个作业到获得输出之间的响应时间要用小时计算。MULTICS的主要目标就是开发这样的一个操作系统，不仅允许多个用户共享CPU的时间(multiuser timesharing)，还让每个用户都可以访问自己的终端。尽管贝尔实验室和通用电气公司最终放弃了这一项目，但MULTICS仍在不少地方得以实际应用。

UNIX的开发始于把日趋没落的MULTICS移植到PDP－7小型计算机上，让这个新操作系统能支持一种新的文件系统。这个新文件系统是UNIX文件系统的第一个版本。这个操作系统，由Ken Thompson开发，支持两个用户，有一个命令解释器和允许新文件系统进行文件操作的一组程序。1970年，UNIX被移植到PDP－11上，经修改后支持更多的用户。从技术上说这就是第一个UNIX版本。

1973年发布的UNIX第四版是由Ken Thompson和Dennis Ritchie用C(由Ritchie那时开发的一种语言)重新编写的。这就让操作系统远离纯汇编语言，并敲开了操作系统可移植的大门。费心想一下这种决定有怎样关键的作用。直至当时，操作系统完全是建立在系统体系结构规范之下，因为汇编语言非常特别，不易移植到其他体系结构中。用C语言对UNIX重写是向更可移植(和可读)的操作系统迈向的第一步，也是让UNIX变得迅速普及的一步。

1974年是UNIX在大学中迅速普及的标记性年份。学术机构开始与贝尔实验室的UNIX系统开发组进行合作，开发出很多具有创新特色的第五版。这一版本可免费获得，并把源代码提供给大学用来教学。1979年，在众多创新、代码简化以及改善可移植性之后，UNIX操作系统第七版(V7)诞生了。这一版本包含了C编译器和著名的B shell命令解释器。

20世纪80年代标志着个人计算机时代的到来。当时工作站只用在商业和大学中。大量UNIX变种从第七版开发而来。这些变种包括Berkley UNIX(BSD)和AT&T UNIX System III和System V，其中BSD是由加利福尼亚大学伯克利分校开发的。每个变种又会演变出其他系统，如NetBSD和OpenBSD（BSD的变种)，以及AIX（IBM的System V的变种)。事实上，UNIX的所有商用变种都来源于System V或BSD。

1991年，Linux出现，那时，UNIX正非常流行，但不适用于PC机上。购买UNIX的价格令人望而却步，的确不是一般用户可以接受的，除非他与大学有关。Linux的实现首次扩充了叫做Minix的操作系统(由Andrew Tanenbaum开发的一个教学用小操作系统)。

随后的几年，Linux内核，与自由软件基金会(Free Software Foundation，FSF)GNU项目所提供的系统软件相结合，使得Linux[⊖]发展成为非常坚实的系统，吸引着黑客以外的人员参与进来。1994年，Linux的第一版发布。从那时开始，Linux迅速成长，对Linux发布版的需求量剧增，要求支持各种体系结构的大学、公司及个人用户的数量不断增加。

1.2 标准和通用接口

通用标准在不同种类的UNIX之间架起一座桥梁。用户决定使用哪种UNIX直接影响到它的移植性，从而影响到它的潜在市场。如果你是一名开发者，很显然，你的程序的市场局限

⊖ 为了确信Linux是FSF的GNU项目所提供的系统软件的组成，Linux也被称为GUN/Linux。

于那些曾经使用这一程序的人，除非你不怕麻烦地移植该程序。标准的产生源于需要一种通用的标准接口规范，使得在一种操作系统上开发的程序在不修订或尽可能少修订的情况下可以在另一种操作系统下运行。各种标准组织都开始为 UNIX 制定规范。POSIX 就是其中的一种，这是由美国电子工程学会(IEEE)制定的一种用于计算机环境的可移植操作系统标准，Linux 就遵从这一标准。

1.3　自由软件和开放源码

Linux 是开放源码软件最成功的案例之一。开放源码软件是这样一种软件，它的源代码可以自由获取，以便任何人都可以修改、阅读和重新发布。与之形成对照的是封闭源码(closed-source)软件，它仅以二进制形式发布。

开放源码允许用户按自己的需要随意开发软件。不过，某些限制会应用于代码，这取决于许可证。这样做的好处是用户决不受限于其他用户曾经开发的东西，因为他们可以自由地修改代码以满足自己的需要。Linux 提供了一个开放的操作系统，任何人都可以对 Linux 操作系统进一步开发并贡献自己的智慧。这就使得 Linux 能快速发展，不管是开发、测试还是文档，其推进速度都是令人惊奇的。

有各种各样的开源许可证：特别说明的是，Linux 是在 GNU 的 GPL(General Public License)版本 2 下发布的。在源代码根的 COPYING 文件中，可以找到该许可证的拷贝。如果你打算修改 Linux 内核，那么最好熟悉这个许可证的术语，这样，你就可以知道自己所做工作是否合法。

围绕自由而开放源码软件的产权有两大主要阵营。自由软件基金会(Free Software Foundation)和开源社区的观点有所不同。自由软件基金会是两大组织中老早出现的组织，他们坚持认为字眼“自由”不仅应当适用于软件，还同样适用于公开演说中。开源社区则认为自由而开放的软件与私有软件有不同的方法论。更多的信息，请查阅 http://www.fsf.org 和 http://www.opensource.org 网站。

1.4　Linux 发布版的快速浏览

我们曾提到，Linux 内核仅仅是通常所说的“Linux ”的一部分。Linux 发布版包含 Linux 内核、各种工具、窗口管理器以及很多其他应用程序。Linux 中使用的很多系统程序都是由 FSF GNU 项目开发和维护的。随着对 Linux 日益流行和需求的增长，把内核和这些工具打包在一起成为一件十分有利的事。公司和社区的人们为一组特定的目标提供一个特定的发布版。我们不必关注过多的细节，只回顾一下到写本书为止的主要 Linux 发布版即可。新 Linux 发布版仍在继续发布着。

大多数 Linux 的发布版都把工具和应用程序组织为头文件和可执行文件群。这种分组就是所谓的包，相对于下载头文件并编译源代码，这也是使用 Linux 发布版的主要优点。参照 GPL，该许可证给了把附加值追加到开放源码软件的自由，例如在代码重新发布时提供一些服务。

1.4.1 Debian

Debian[㈠]是一种 GNU/Linux 操作系统。与其他发布版类似，大多数应用程序和工具都来自 GUN 软件和 Linux 内核。Debian 具有较好的包管理系统 apt (advanced packaging tool)。Debian 的主要缺点是开始的安装过程，Linux 新手会感到迷惑。Debian 并不局限于某个公司，它是由志愿者社团开发的。

1.4.2 Red Hat/Fedora

Red Hat[㈡](公司)是开源软件开发平台的主要参与者。Red Hat 以前只有一种 Linux 发布版，2002 年以后，它提供了两种独立的发布版：Red Hat 企业级 Linux 和 Fedora Core。Red Hat 企业级 Linux 的目标是为商业、政府或者其他行业提供稳定而得到支持的 Linux 环境，而 Fedora Core 的目标是个人用户和爱好者。两个发布版之间的主要差异是与新特性相对的稳定性。Fedora 会有较新但不很稳定的代码包含在发布版中。Red Hat 在美国似乎是 Linux 企业级版本的首选。

1.4.3 Mandriva

Mandriva Linux[㈢](以前叫做 Mandrake Linux)是 Red Hat Linux 的易安装版，但之后脱胎为一个独立的发布版，用于 Linux 的个人用户。Mandriva Linux 的主要特点是系统配置和安装都较容易。

1.4.4 SUSE

SUSE Linux[㈣]是 Linux 舞台的另一个主要参与者。SUSE 主要被用在商业、政府、工业和个人用户，它的主要特点是它的安装和管理工具 Yast2。SUSE 在欧洲似乎是 Linux 企业级版本的首选。

1.4.5 Gentoo

Gentoo[㈤]是销售中的 Linux 新发布版，它已经赢得了众多赞许。Gentoo Linux 的主要特点就是其所有包都是根据自己机器的特殊配置从源代码编译而来的，这可以通过 Gentoo portage 系统来完成。

1.4.6 Yellow Dog

Yellow Dog(黄狗)Linux[㈥]是基于 PPC 的 Linux 发布版的主要参与者之一。尽管新近描述的

㈠ http://www.debian.org。

㈡ http://www.redhat.com。

㈢ http://www.mandriva.com/。

㈣ http://www.novell.com/linux/suse/。

㈤ http://www.gentoo.org/。

㈥ http://www.yellowdoglinux.com/。