带你深入Linux的世界，
让你的嵌入式和运维开发更上一层楼，对Linux系统的运用成竹在胸。

Broadview
www.broadview.com.cn

深入Linux内核架构与底层原理

刘京洋 韩方 著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

深入 Linux

内核架构与底层原理

刘京洋 韩方 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书主要描述 Linux 系统的总体框架和设计思想，包含很多可以直接操作的实例，目的是希望读者对 Linux 系统背后的逻辑有一个全面的了解。本书力求贴近实际的工作使用，在比较核心且常用的技术点有更加深入的解释，对实际使用 Linux 系统工作大有裨益。

本书共 13 章，其中第 1~3 章是总览，第 4~13 章是分领域阐述。第 1~3 章总体介绍 Linux 的基本知识；第 4 章以 Linux 系统的启动开始深入叙述；第 5 章是 Linux 系统运行中使用者最常接触到的进程概念，重点介绍进程的原理；第 6 章是 Linux 内核的内存管理方法与用户端使用内存的底层方法，即重点介绍 glibc 底层到内核之间的内存管理过程；第 7~13 章分别是关于安全、网络、总线与设备变动、二进制、存储、虚拟化与云、硬件专用子系统的内容。这些子系统都是 Linux 系统运行中非常重要的领域，是深入理解 Linux 系统原理不可或缺的知识补充。

图书在版编目（CIP）数据

深入 Linux 内核架构与底层原理 / 刘京洋，韩方著. —北京：电子工业出版社，2017.11
ISBN 978-7-121-32290-7

Ⅰ. ①深… Ⅱ. ①刘… ②韩… Ⅲ. ①Linux 操作系统 Ⅳ. ①TP316.85

中国版本图书馆 CIP 数据核字（2017）第 176716 号

策划编辑：黄爱萍
责任编辑：徐津平
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：787×980　1/16　印张：24.75　字数：453 千字
版　　次：2017 年 11 月第 1 版
印　　次：2017 年 11 月第 1 次印刷
定　　价：89.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）51260888-819，faq@phei.com.cn。

推 荐 序

Linux 操作系统在超级计算机、互联网服务、桌面系统、移动和嵌入式设备等领域使用广泛，相关的从业人员和兴趣爱好者一直对 Linux 的理论和实践有较大需求。本人作为互联网领域的从业人员之一，非常荣幸可以提前阅读书中的内容，发现不同于市面上常见的 Linux 书籍，本书的内容从内核层面出发，对各个子系统的设计、实现和演化进行了梳理，并结合作者多年的亲身实践。以下部分是本人阅读后，希望与读者分享的一些感受。

第一个特点是**解释透彻**。Linux 发展至今已经超过 25 年，源代码融合了不同时期的演进和变化，因此回顾当时的背景，有助于更清晰地了解代码作者的意图和目标。以本书第 4 章“Linux 系统的启动”为例子，介绍了 BIOS 的局限性，如何向 EFI 演进的历史，另外还重点分析了 initrd 过渡根文件系统的来龙去脉，让本人对 Linux 启动的过程有更全面而深入的认识。又例如第 12 章“虚拟化与云”中的第二节，介绍了一段比较近的融合文件系统的历史，Docker 最初使用的 AUFS 逐渐过渡到合并到内核的 Overlay。

第二个特点是**实践性强**。在技术领域，实践往往能加快加深对相关概念的理解，本书有不少例子适合当作实验，感兴趣的读者可在单机环境或者虚拟机环境完成。例如上面提到的 initrd 文件系统的例子，书中比较完整地介绍了几种可行的制作方法。而在介绍 cgroup 的章节中，作者介绍了不同子系统可以限制的资源和效果，感兴趣的读者可以通过操作 cgroup 文件系统来观察实现的效果。再介绍第 8 章“网络”

中的一个有趣的“六次握手”实验，这里可以看到一些很少见到的两端同时发起连接的 TCP 状态变化。

第三个特点是**指路明灯**。Linux 内核的子系统和模块非常多，覆盖的应用范围也很广阔，面面俱到显然是不现实的。作者希望更多地展示代码背后的思想，以及作者思虑后的理解，这比单纯的技术讲解更有营养价值，同时也鼓励读者在阅读时形成自己的见解，学会自己查阅相关的技术资料。例如，第 6 章“Linux 内存管理”中介绍了内存回收算法 PFRA，本人阅读前并不是特别清楚匿名页和命名页的不同处理流程，以及系统在内存低水位的行为，阅读后形成线索，可搜索出更多相关的资料。

第四个特点是**与时俱进**。近几年，业界利用 Linux 构建很多热点应用，本书在很多方面覆盖了 Linux 较新的功能，对从业者有较大帮助。例如，容器技术在应用服务上非常火热，本书在第 12 章“虚拟化与云”中对 cgroup 资源隔离和命名空间有基本的介绍。I/O 方面，本书介绍了能大幅改善性能的用户态 I/O，包括目前高性能网络中用到的 DPDK。在第 7 章“安全”中，还介绍了内核的 eBPF 虚拟机，很多新的内核调试工具、审计工具和高性能包处理都依赖这个机制。

总的来说，本人阅读后收获颇丰，对工作也有积极帮助，希望其他读者也能从中获取价值。最后，本人也是一名开源爱好者，感谢作者的辛勤付出，编写出一本内容详尽的 Linux 著作，希望 Linux 和其他开源社区发展越来越好。

李文俊

前　言

要想深入研究并使用 Linux 内核，首先要知道 Linux 内核提供了什么，又能做到什么。很多初学者一进入公司就开始使用 Linux 内核开发内核模块，无论是使用通信方式、内存接口还是设备接口，都是早已被淘汰的内容。因为他们通常直接在网络上搜索一些很早之前发布的内容来指导自己如何完成开发工作，但他们手中却是最先进的内核代码。还有很多直接编写内核模块的人在嵌入式公司使用老版本的内核进行工作，虽然他们可能对内核之后的发展一无所知，但是他们能够一下子抓住主干，主干永远是在老版本的内核中就存在的东西。

很多刚入行的程序员认为自己能够征服一切，稍微在网上检索一下 Linux 的内容，就可以上手使用了。虽然写出可以用的程序不需要太多的知识积累，但是这么做相当于在信息不充分的情况下做决策。虽然一切操作系统理论的学习都不如实际去编写几行代码，但是理论又是十分重要的，因为它能够让经验升华成积累。

本书解释了 Linux 内核提供了什么，以及 Linux 系统底层是如何使用内核的。如果你对本书某一部分感兴趣，那么在深入阅读该部分的代码之前应先对该内容进行系统的学习，当你对内核系统有一个整体的把握时，方可挥洒自如。

本书的读者对象是有一定 Linux 基础的程序员，或者是有一定经验的嵌入式开发人员和运维人员。阅读本书像喝水一样，可轻松获得知识内容。若阅读本书遇到相对冷门的技术细节时，有兴趣的读者可以自行查阅其他相关资料。例如当列举文件系统的种类时提到 exofs，书中不会过多解释这个名词，因为大部分用户只关注它是文件系统的一种。

在学习 Linux 内核，阅读相关图书时候限定版本是不必要的，因为即使版本变化，原理仍旧可用。本书也会注明某个技术点之前是什么样的，现在是什么样的，未来可能是什么样的。人们更希望了解整个内核框架的内容，以及一些重要细节的深层原理。本书就将重点放在这两方面内容上，而并不局限于内核的版本，尽可能以最终被选择的解决方案作为实验重点。也就是说，本书所涉及的内核版本都比较新，但是也会观察从老版本到新版本过渡时内核在功能上的变化，比如 ip rule 命令在新版本中去掉了 reject 等 action。但是老版本的设计对于整体理解架构很有帮助，我们的根本目的是用实现抽象出概念，本书讲解的所有案例几乎都使用了占据较大市场份额的 Ubuntu。

感谢韩方，他对本书的出版起到了提纲挈领的作用，若没有他的帮助，我一定会被淹没在一堆技术细节中而不知道如何选择。他编写并且修改了部分章节，概览性质的图书最需要高屋建瓴的能力和丰富的经验，韩方在这方面非常强。

由于时间仓促，加之水平有限，书中的缺点和不足之处在所难免，敬请读者批评指正。

刘京详

2017 年 10 月

目　录

第 1 章

总览

1.1 简介

在 Linux 内核出现之前出现过很多优秀的内核，甚至同时期直至今日，那些竞争关系的内核仍然存在。2007 年我刚学习 Linux 时，还在犹豫是否要学习和使用 FreeBSD，现在我绝对不会有疑惑了。可以说 UNIX 是 Linux 发展壮大的原因，Minix 是 Linux 发展的方法。Linux 从对 Minix 的深入研究发源而来，但是实际地推出大量借鉴了 UNIX 的交互方式和技术模式。UNIX 本来也完全可以压倒性地压制 Linux，因为那时的 UNIX 内核有多个封闭版本，并且都是商业运行，很多非常强大。Linux 之所以能够后来逐步超越 UNIX，是因为 Linux 的开源属性和 Torvalds 的杰出领导。

直到今天一个软件想要获得最广泛的应用，开源仍旧是不二手段。通过这种方式能够让自己的软件取得比更加友好强大的商业软件更大的市场空间。但是代价就是难以找到盈利模式。UNIX 就像一个创新发源地，很多需要花费巨资的 Linux 的标准化工作，例如 POSIX 都由 UNIX 背后的企业去完成，一些新奇的技术也是由在商业系统研究的过程中发表的 paper，被 Linux 直接借鉴实现。近年 Android 的发展也让 Linux 从谷歌获得了大量的改进和修复（最典型的是 cgroup）。久而久之就形成了 Linux 独特的价值观：求全第一，求精第二。

Plan9 是由贝尔实验室开发的系统，参与阵容可谓豪华：Rob Pike（现在在 Google

工作，负责 Go 语言的开发）；Ken Thompson（C 语言和 UNIX 创始人）；Dennis Ritchie（C 语言和 UNIX 创始人）；Brian Kernighan（awk 之父）；Doug Mcilroy（UNIX 管道提出者，UNIX 开发参与者）。我在使用 Linux 时最大的震撼就是一切皆文件的思想，然而这在 Plan9 中已然是价值观上的定海神针。Plan9 中最本质的思想是“一切皆是文件”，甚至 CPU 是一个文件；内存是一个文件；网络是一个文件，任何的东西都是一个文件。这在 Linux 中已经尽量做到这一点，在 Plan9 中被极度地强化。

Linux 和 UNIX 是一个多用户分时操作系统，就是多个用户共享一个操作系统资源。不管是 CPU、内存、网络，都需要通过调度器分配调度。比如 A 机器的文件需要使用 B 机器的 CPU 来处理，方法就只有通过某种协议，将 A 机器的文件下载到 B 机器中，然后 B 机器处理完以后再回传到 A 机器中。Plan 9 的“一切皆是文件”看起来很好的解决了这个问题。A 机器想使用 B 机器的 CPU，只需要将 B 机器的 CPU 挂载到 A 机器的 CPU 的文件中就能完成这个需求了。当然，两个机器之间也有一个协议“9P”来进行文件挂载和表示，但是这个操作对上层的操作系统来说已经是透明的了。

Plan9 是一个分布式操作系统，它能把网络上一切资源当作文件来进行使用，这其实就是云的概念了。但是看起来好的东西在实际使用中不一定好，很多时候人们会更倾向于使用 socket，就像它本可以用 Python 语言一句话完成一个 socket，但是它在某些极端情况下还得用 C 语言的 socket，因为那样更加可控并且效率高。当你挂载了远端的 CPU 到本机时，用起来像用本地的 CPU 一样，这看起来是优点，但同时也是缺点，使用这个系统的人会莫名其妙地发现自己的程序很慢，但是他很难想到是被调度到远端的 CPU 上去执行了，如果想要加速程序运行速度，就得深入地研究 CPU 挂载的流程和原理。实际上不是那么方便，这有时候就是对失误的一个很好的阻碍。

我们做云系统的时候就会发现，很多时候为了可靠性和稳定性都会在一定程度上牺牲易用性。一个操作系统最重要的素质就是稳定性，否则任何的上层进程出了问题，还得去找系统级的原因，那样就会极大地阻碍这个系统的实际有效应用。

除了 Plan 9 之外，Linux 系统还有很多重量级的竞争对手：Haiku、BeOS、Amiga、OS/2、Arthur、XTS-400、Infemo、Symbian、Palm OS 等。Linux 系统目前在嵌入式和服务器领域取得的成功并不是因为它一开始就是这个定位，而是只有这两个大方向显著适合 Linux 系统，因为 Linux 系统什么都能做。而其他的竞争对手通常都有共同的特点，那就是专业或者商业。比如 BeOS 为多媒体而专门设计得十分精良，还差一点被苹果公司用作桌面系统，它的定位可以说是非常精准，但是市场无数次

的证明了胜出的不一定是好的。但是其派生的开源系统 Haiku 延续它发展了一段时间，但是也失败了。就连微软、IBM 联合重金研发，诞生的当今软件工程圣典《人月神话》的 OS/2 项目也迅速归于失败。Inferno 操作系统继承自 Plan 9、Amiga 系统同期拥有不亚于苹果的技术实力，也都因为过度依赖于某个公司的开发，经营不善而没落。我们熟知的 Palm OS 和 Symbian 也都饱尝了封闭拒绝进步带来的恶果。1990—2000 年是一个时代，这个时代产生了许多未来操作系统的种子选手。

信息行业几乎已经证明了一件事，在一个领域几乎只能容纳一两家巨头。那时的先驱者不一定认识到这个问题。他们或者是因为经营问题、销售问题、定位问题、技术问题，又或者是因为资金问题，他们都失败了。Linux 系统走了完全不同的路，这条路可以说是操作系统选手里面 Minix 最早尝试的，但是有完整的源代码。无疑是 Minix 的这种模式直接导致了 Linux 的诞生，其他操作系统走过的弯路，Linux 不必再走。Linus 本人是一个有情怀的技术人，他没有选择试图成立公司，他可能失去的是一个伟大的公司，但是他却收获了一个伟大的产品，并且极大地促进了整个人类信息技术的进步。

Linux 系统是一个工具，也是一个现象、一个平台。它很可能是最近 20 年对信息产业贡献最大的开源项目。路由器、手机、监控系统、互联网的服务器，甚至桌面系统都逐渐的在应用 Linux 系统，甚至除了桌面系统，在大部分的信息系统里，Linux 系统已经几乎成为了唯一的选择。大部分的应用场景都是使用 Linux 作为一个操作系统平台，而在之上运行企业自己的应用。根据这个应用对底层的需求层次不同，开发者对 Linux 知识了解程度的需求是截然不同的。纯粹逻辑的应用是几乎完全不需要感知到 Linux 的存在的。编写逻辑，编译部署到 Linux 系统上就可以运行。但是包括运维、嵌入式，尤其是涉及更高的性能要求或者稳定性要求等时，就需要借助 Linux 内核或者系统底层提供的机制了。

Linux 内核提供的机制比较常用的部分，在用户端基本会有底层库进行封装提供，甚至一些内核没有提供的能力，底层库也会提供。所以当你需要底层知识时，可能使用的是底层库提供的机制，也可能是底层库提供不了的，此时就由内核直接提供。通常情况下底层库都需要尽可能地封装内核提供的功能，但是由于 Linux 内核的特性，它尽可能地集中了功能，并且直接通过文件系统接口暴露，这就导致了例如/proc、/dev 等文件系统的使用可以非常方便地直接调用到内核的功能，底层库也就没有必要封装了（虽然大部分它们也都封装了）。

所以当我们深入到 Linux 时，通常会同时接触到底层库和内核的用户空间接口。