

HZ BOOKS
华章科技

历时8年，三次重构，内容愈加炉火纯青。

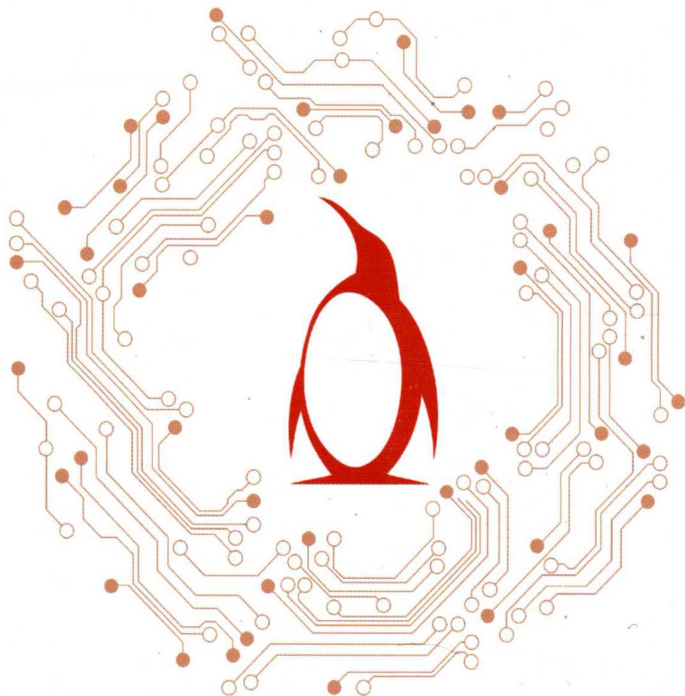
全部代码更新至全新的Linux 4.0版本。

全面讲解ARM Linux新版本内核架构，如设备树等。

不仅仅注重知识和程序的讲解，更注重程序的思想、演变、架构和算法。



嵌入式Linux



ARM

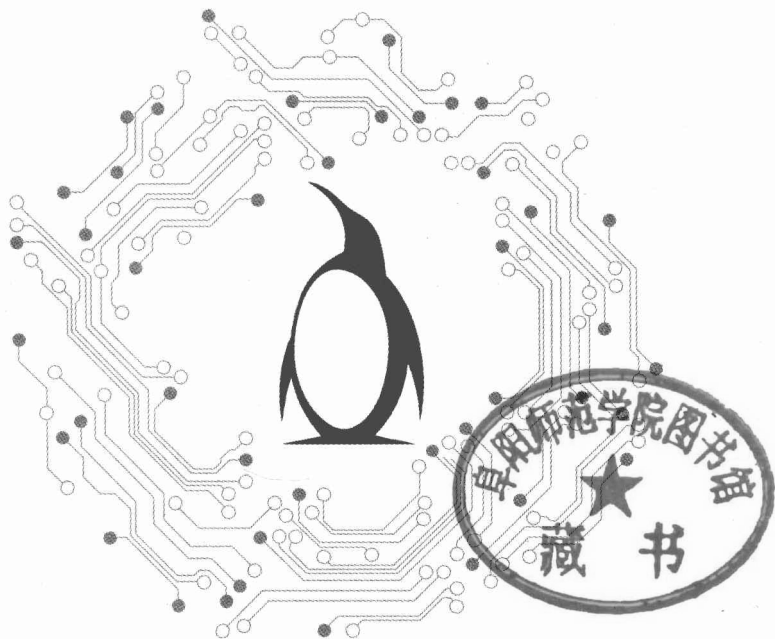
Linux设备驱动开发详解

基于最新的Linux 4.0内核

宋宝华 编著



机械工业出版社
China Machine Press



ARM

Linux设备驱动开发详解

基于最新的Linux 4.0内核

宋宝华 编著



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

Linux 设备驱动开发详解: 基于最新的 Linux 4.0 内核 / 宋宝华编著. —北京: 机械工业出版社, 2015.7

(电子与嵌入式系统设计丛书)

ISBN 978-7-111-50789-5

I. L… II. 宋… III. Linux 操作系统—程序设计 IV. TP316.89

中国版本图书馆 CIP 数据核字 (2015) 第 150963 号

Linux 设备驱动开发详解: 基于最新的 Linux 4.0 内核

出版发行: 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: 张国强

责任校对: 殷虹

印刷: 三河市宏图印务有限公司

版次: 2015 年 8 月第 1 版第 1 次印刷

开本: 186mm × 240mm 1/16

印张: 39.75

书号: ISBN 978-7-111-50789-5

定价: 89.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88379426 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzit@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

赞 誉

十多年前，我在海外一家路由器公司从事底层软件开发时，一本《Linux Device Driver》(LDD) 使我受益匪浅。近年来，我在从事基于 ARM 的嵌入式操作系统教学时发现，很多 Linux 设备驱动中的新技术，比如 Device Tree、sysfs 等，在 LDD3 中均没有涉及。而市面上的翻译书晦涩难懂，有的还不如英文原书好理解。宋宝华是我尊敬的技术人员，十年如一日，在 Linux 内核及设备驱动领域潜心耕耘，堪称大师。本书无论从汉语的遣词造句，案例的深入浅出，还是对前沿技术的掌握，对难点技术丝丝入扣的分析，都体现出了强烈的“工匠精神”，堪称经典，值得推荐。

——Xilinx 前大中华区大学计划经理、慕容信 CEO 谢凯年

设备驱动程序是连接计算机软件和硬件的纽带和桥梁，开发者在嵌入式操作系统的开发移植过程中，有将近 70% ~ 80% 的精力都用在了驱动程序的开发与调试方面。这就对设备驱动程序开发人员提出了极高的要求。开发者不仅要同时具备软件和硬件的知识和经验，而且还要不断地学习、更新自己，以便跟上嵌入式系统日新月异的发展。研究前人的总结和动手实践是不断提高自己的有效途径。虽然市面上已经有多种设备驱动的书籍，但本书在总结 Linux 设备驱动程序方面仍然非常具有特色。它用理论联系实际，尤其是提供了大量的实例，对读者深入地理解并掌握各种驱动程序的编写大有裨益。

——飞思卡尔半导体(中国)有限公司数字网络软件技术方案部总监 杨欣欣博士

一位优秀的设备驱动开发工程师需要具备多年嵌入式软件和硬件开发经验的积累，本书针对 Linux 设备驱动开发相关的设计思想、框架、内核，深入浅出，结合代码，从理论到实践进行重点讲解。毫无疑问，本书可谓一把通向设备驱动大师殿堂之门的金钥匙，它将激发你的味蕾，带你“品尝”嵌入式设备驱动开发这道“美味佳肴”，掩卷沉思，意味深长。

——ARM 中国在线社区经理 宋斌

作者长期从事嵌入式 Linux 开发和教学工作，擅长 Linux 驱动开发，并跟踪开源软件和嵌入式处理器技术的最新发展，撰写本书，书中内容新鲜实用。作者针对 ARM 和移动便携设备的兴起，在书中添加了 ARM Linux 设备树和 Linux 电源管理系统架构及驱动的内容，书中关于 Linux 设备驱动的软件架构思想的章节也很有特色。

——中国软件行业协会嵌入式系统分会副理事长 何小庆

推荐序一

技术日新月异，产业斗转星移，滚滚红尘，消逝的事物太多，新事物的诞生也更迅猛。众多新生事物如灿烂烟花，转瞬即逝。当我们仰望星空时，在浩如烟海的专业名词中寻找，赫然发现，Linux 的生命力之旺盛顽强，斗志之昂扬雄壮，令人称奇。它正以摧枯拉朽之势迅速占领包括服务器、云计算、消费电子、工业控制、仪器仪表、导航娱乐等在内的众多应用领域，并逐步占据许多 Wince、VxWorks 的传统嵌入式市场。

Linux 所及之处，所向披靡。这与 Linux 的社区式开发模式，迅速的迭代不无关系。Linux 每 2 ~ 3 月更新一次版本，吸纳新的体系架构、芯片支持、驱动、内核优化和新特性，这使得 Linux 总是能够在第一时间内迎合用户的需求，快速地适应瞬息万变的市场。由 Linux 以及围绕着 Linux 进行产品研发的众多企业和爱好者构成了一个庞大的 Linux 生态圈。而本书，无疑给这个庞大的生态圈注入了养料。

然而，养料的注入应该是持续不断的。至今，Linux 内核的底层 BSP、驱动框架和内核实现发生了许多变更，本书涵盖了这些新的变化，这将给予开发者更多新的帮助。内核的代码不断重构并最优化，而本书也无疑是一次重大的重构。

生命不息，重构不止。

周立功

推荐序二

在翻译了《Understanding the Linux Kernel》和《Linux Kernel Development》这两本书后，每当有读者询问如何学习 Linux 内核时，我都不敢贸然给出建议。如此庞大的内核，各个子系统之间的关系错综复杂，代码不断更新和迭代，到底该从何入手？你的出发点是哪里？你想去的彼岸又是哪里？相应的学习方法都不同。

一旦踏入 Linux 内核领域，要精通 Linux 内核的精髓，几乎没有捷径可走。尽管通往山顶的路有无数条，但每条路上都布满荆棘，或许时间和毅力才是斩荆披棘的利器。

从最初到现在，Linux 内核的版本更新达上千个，代码规模不断增长，平均每个版本的新增代码有 4 万行左右。在源代码的 10 个主要子目录（arch、init、include、kernel、mm、IPC、fs、lib、net、drivers）中，驱动程序的代码量呈线性增长趋势。

从软件工程角度来看内核代码的变化规律，Linux 的体系结构相对稳定，子系统数变化不大，平均每个模块的复杂度呈下降趋势，但系统整体规模和复杂性分别呈超线性和接近线性增长趋势。drivers 和 arch 等模块的快速变化是引起系统复杂性增加的主因。那么，在代码量最多的驱动程序中，有什么规律可循？最根本的又是什么？

本书更多的是关于 Linux 内核代码背后机理的讲解，呈现给读者的是一种思考方法，让读者能够在思考中举一反三。尽管驱动程序只是内核的一个子系统，但 Linux 内核是一种整体结构，牵一发而动全局，对 Linux 内核其他相关知识的掌握是开发驱动的基础。本书的内容包括中断、定时器、进程生命周期、uevent、并发、编译乱序、执行乱序、等待队列、I/O 模型、内存管理等，实例代码也被大幅重构。

明代著名的思想家王阳明有句名言“知而不行，是为不知；行而不知，可以致知”。因此在研读本书时，你一定要亲身实践，在实践之后要提升思考，如此，你才可以越过代码本身而看到内核的深层机理。

陈莉君

西安邮电大学

前 言

Linux 从未停歇前进的脚步。Linus Torvalds，世界上最伟大的程序员之一，Linux 内核的创始人，Git 的缔造者，现在仍然在没日没夜地合并补丁、升级内核。做技术的人，从来没有终南捷径，拼得就是坐冷板凳的傻劲。

这是一个连阅读都被碎片化的时代，在这样一个时代，人们趋向于激进、浮躁，内心的不安宁使我们极难静下心来研究什么。我见过许多 Linux 工程师，他们的简历上写着“精通”Linux 内核，有多年的工作经验，而他们的“精通”却只是把某个寄存器从 0 改成 1，从 1 改成 0 的不断重复；我也见过许多 Linux 工程师，他们终日埋头苦干，敲打着自己的机器和电路板，却从未冷静下来思考，并不断重构和升华自己的知识体系。

这是要把“牢底”坐穿的程序员，这样“忙忙碌碌”的程序员，从来都不算是好程序员。

对于优秀的程序员，其最优秀的品质是能够心平气和地学习与思考问题，透析代码背后的架构、原理和设计思想。没有思想的代码是垃圾代码，没有思想的程序员，只是在完成低水平重复建设的体力活。很多程序员从不过问自己写的代码最后在机器里面是怎么跑的，很多事情莫名其妙地发生了，很多 bug 莫名其妙地消失了……他们永远都在得过且过。

由此，衍生出了本书的第一个出发点，那就是带给读者更多关于 Linux 开发思想的讲解，帮助读者奠定根基。本书呈现给读者的更多的是一种思考方法，而不是知识点的简单罗列。

本书除对基础理论部分进行了详细的讲解外，还加强了对驱动编程所涉及的 Linux 内核最底层机理的讲解，内容包括中断、定时器、进程生命周期、uevent、并发、编译乱序、执行乱序、等待队列、I/O 模型、内存管理等。这些知识点非常重要，是真正证明程序员理解了 Linux 的部分内容，程序员只有打好根基，才能游刃有余。

本书没有大量描述各种具体驱动类型的章节，如 Sound、PCI、MTD、tty 等，而将更多的焦点转移到了驱动编程背后的内核原理，并试图从 Linux 内核的上百个驱动子系统寻找出内部规律，以培养读者举一反三的能力。

Linux 内核有上百个驱动子系统，这一点从内核的 drivers 子目录中就可以看出来：

| | | | | | | | |
|---------------|-------------|------------|-----------|-----------------|------------|-------------|----------|
| accessibility | clocksource | fmc | irqchip | mmc | platform | sbus | usb |
| acpi | connector | gpio | isdn | modules.builtin | pnP | scsi | uwb |
| amba | coresight | gpu | Kconfig | modules.order | power | sfi | vfio |
| android | cpufreq | hid | leds | mtD | powercap | sh | vhost |
| ata | cpuidle | hsi | lguest | net | pps | sn | video |
| atm | crypto | hv | macintosh | nfc | ps3 | soc | virt |
| auxdisplay | dca | hwmon | mailbox | ntb | ptp | spi | virtio |
| base | devfreq | hwspinlock | Makefile | nubus | pwm | spmi | vlynq |
| bcma | dio | i2c | ncb | of | rapidio | ssb | vme |
| block | dma | ide | nd | oprofile | ras | staging | w1 |
| bluetooth | dma-buf | idle | media | parisc | regulator | target | watchdog |
| built-in.o | edac | io | memory | parport | remoteproc | tc | xen |
| bus | eisa | infiniband | memstick | pci | reset | thermal | zorro |
| cdrom | extcon | input | message | pcmcia | rpmsg | thunderbolt | |
| char | firewire | iommu | mfd | phy | rtc | tty | |
| clk | firmware | ipack | misc | pinctrl | s390 | uio | |

好吧，傻子才会一个目录一个目录地去看，一个目录一个目录地从头学起。我们势必要寻找各种驱动子系统的共性，摸索规律。在本书中，我们将更多地看到各驱动子系统的类比，以及驱动子系统的层次化设计。

技术工作从来都不能一劳永逸。世界变化得太快，当前技术革新的速度数倍于我们父辈、祖辈、祖祖辈经历过的任何时代。证明你是“真球迷”还是“伪球迷”的时候到了，这个时代是伪程序员的地狱，也是真程序员的天堂。

从浩如烟海的知识体系、不断更新的软件版本中终生学习，不断攻克一个个挑战，获取新养分，寻找新灵感，这实在是黑暗的码农生涯中不断闪现的璀璨光芒。

Linux 的内核版本不断更新，出现了 Linux 3.0、Linux 3.1、Linux 3.2、…、Linux 3.19、Linux 4.0、Linux 4.1，变化的是软件的架构，不变的是 Linus 的热情。

这无疑也是本书的第二个出发点，更新 Linux 驱动编程的知识体系以迎合最新的时代需求。因此，本书有大量关于设备树、ARM Linux 移植、Linux 电源管理、GPIO、时钟、定时器、pinmux、DMA 等内容。我们的操作平台也转移到了 QEMU 模拟的 4 核 Cortex-A9 电路板上，书中的实例基本都转移到了市面流行的新芯片上。

最近两三年，老是听许多程序员抱怨，市面上缺乏讲解新内核的资料、缺乏从头到尾讲解设备树的资料，但是我想说，这实在不是什么难点。难点仍然是本书基于第一个出发点要解决的问题，如果有好的基础，以优秀程序员极强的学习能力，应该很快就可以掌握这些新知识。机制没有变，变化的只是策略。

因此学习能力也是优秀程序员的又一个重要品质。没有人生下来就是天才，良好的学习能力也是通过后天的不断学习培养的。可以说，学得越多的人，学新东西的速度一定越快，学习能力也变得越强。因为，知识的共通性实在太多。

读者在阅读本书时，不应该企图把它当成一本工具书和查 API 的书，而是应该把它当作一本梳理理论体系、开发思想、软件架构的书。唯如此，我们才能适应未来新的变化。

时代的滚滚车轮推动着 Linux 内核的版本不断向前，也推动着每个人的人生。红尘滚滚，

我不去想是否能够成功，
既然选择了远方，
便只顾风雨兼程。

最后，本书能得以出版，要感谢带领我向前的人生导师和我的众多小伙伴，他们或者在我人生的关键时刻改变了我，或者给我黑暗的程序生涯带来了无尽的快乐和动力。我的小伙伴，他们力挺我、鼓励我，也辱骂我、奚落我，这些都是真挚的友情。

谨以此书，致以对杨平先生、何昭然、方毅伟、李华毅、宋志武、杜向龙、叶祥振、刘昊、王榕、何晔、王立赛、曾过、刘永生、段丙华、章君义、王文琪、卢鹏、刘涛、徐西宁、吴赫、任桥伟、秦龙廷、胡良兵、张家旺、王雷、Bryan Wu、Eric Miao、Cliff Cai、Qipan Li、Guoying Zhang、陈健松、Haoyu Zhong、刘洪涛、季久峰、邴杰、孙志忠、吴国举、Bob Liu、赵小吾、EJ Zhao、贺亚锋、刘仕杰、Hao Yin 等老师和小伙伴的深深感激；谨以此书，致对我的父母大人、老婆大人、兄长和姐姐、伟大丈母娘的深深感激，本书的写作时间超过一年，其过程是一种巨大的肉体和精神折磨，没有他们的默默支持和不断鞭策，本书是不可能完成的；谨以此书，对为本书做出巨大贡献的编辑、策划老师，尤其是张国强老师致以深深的感激！

由于篇幅的关系，我没有办法一一列举我要感激的所有人，但是，这些年从你们那里获得的，远远大于我付出的，所以，在内心深处，唯有怀着对你们的深深感恩，不断前行。岁月如歌，吾歌狂行。

全书结构

本书首先介绍 Linux 设备驱动的基础。第 1 章简要地介绍了设备驱动，并从无操作系统的设备驱动引出了 Linux 操作系统下的设备驱动，介绍了本书所基于的开发环境。第 2 章系统地讲解了 Linux 驱动工程师应该掌握的硬件知识，为工程师打下 Linux 驱动编程的硬件基础，详细介绍了各种类型的 CPU、存储器和常见的外设，并阐述了硬件时序分析方法和数据手册阅读方法。第 3 章将 Linux 设备驱动放在 Linux 2.6 内核背景中进行讲解，说明 Linux 内核的编程方法。由于驱动编程也在内核编程的范畴，因此，这一章实质是为编写 Linux 设备驱动打下软件基础。

其次，讲解 Linux 设备驱动编程的基础理论、字符设备驱动及设备驱动设计中涉及的并发控制、同步等问题。第 4、5 章分别讲解 Linux 内核模块和 Linux 设备文件系统；第 6 ~ 9 章以虚拟设备 `globalmem` 和 `globalfifo` 为主线，逐步给其添加高级控制功能；第 10、11 章分

别阐述 Linux 驱动编程中所涉及的中断和定时器、内核和 I/O 操作处理方法。

接着，剖析复杂设备驱动的体系结构以及块设备、网络设备驱动。该篇讲解了设备与驱动的分离、主机控制器驱动与外设驱动的分离，并以大量实例（如 input、tty、LCD、platform、I²C、SPI、USB 等）来佐证。其中第 12 章和第 17 章遥相呼应，力图全面地展示驱动的架构。Linux 有 100 多个驱动子系统，逐个讲解和学习都是不现实的，授人以鱼不如授人以渔，因此我们将更多的焦点放在了架构讲解方面，以便读者可以举一反三。

本书最后 4 章分析了 Linux 的设备树、Linux 移植到新的 SoC 上的具体工作以及 Linux 内核和驱动的一些调试方法。这些内容，对于理解如何从头开始搭建一个 Linux，以及整个 Linux 板级支持包上上下下的关系尤为重要。

另外，本书的主要代码都引用自 Linux 源代码，为保留原汁原味，均延用了代码的英文注释，而其他非引用的代码则使用了中文注释或无注释，特此说明。

本书配套的相关素材和代码，读者均可从与本书相关的微信公众号中获得，相关公众号

是：Linuxer，欢迎扫描二维码



宋宝华

2015 年 4 月于上海浦东

目 录

| | |
|----------------------------------|-----------|
| 赞誉 | |
| 推荐序一 | |
| 推荐序二 | |
| 前言 | |
| 第 1 章 Linux 设备驱动概述及开发环境构建 | 1 |
| 1.1 设备驱动的作用 | 1 |
| 1.2 无操作系统时的设备驱动 | 2 |
| 1.3 有操作系统时的设备驱动 | 4 |
| 1.4 Linux 设备驱动 | 5 |
| 1.4.1 设备的分类及特点 | 5 |
| 1.4.2 Linux 设备驱动与整个软硬件系统的关系 | 6 |
| 1.4.3 Linux 设备驱动的重点、难点 | 7 |
| 1.5 Linux 设备驱动的开发环境构建 | 8 |
| 1.5.1 PC 上的 Linux 环境 | 8 |
| 1.5.2 QEMU 实验平台 | 11 |
| 1.5.3 源代码阅读和编辑 | 13 |
| 1.6 设备驱动 Hello World: LED 驱动 | 15 |
| 1.6.1 无操作系统时的 LED 驱动 | 15 |
| 1.6.2 Linux 下的 LED 驱动 | 15 |
| 第 2 章 驱动设计的硬件基础 | 20 |
| 2.1 处理器 | 20 |
| 2.1.1 通用处理器 | 20 |
| 2.1.2 数字信号处理器 | 22 |
| 2.2 存储器 | 24 |
| 2.3 接口与总线 | 28 |
| 2.3.1 串口 | 28 |
| 2.3.2 I ² C | 29 |
| 2.3.3 SPI | 30 |
| 2.3.4 USB | 31 |
| 2.3.5 以太网接口 | 33 |
| 2.3.6 PCI 和 PCI-E | 34 |
| 2.3.7 SD 和 SDIO | 36 |
| 2.4 CPLD 和 FPGA | 37 |
| 2.5 原理图分析 | 40 |
| 2.6 硬件时序分析 | 42 |
| 2.6.1 时序分析的概念 | 42 |
| 2.6.2 典型的硬件时序 | 43 |
| 2.7 芯片数据手册阅读方法 | 44 |
| 2.8 仪器仪表使用 | 47 |
| 2.8.1 万用表 | 47 |
| 2.8.2 示波器 | 47 |
| 2.8.3 逻辑分析仪 | 49 |

| | | | |
|----------------------------------|-----------|--|------------|
| 2.9 总结 | 51 | 4.6 导出符号 | 99 |
| 第3章 Linux 内核及内核编程 | 52 | 4.7 模块声明与描述 | 100 |
| 3.1 Linux 内核的发展与演变 | 52 | 4.8 模块的使用计数 | 100 |
| 3.2 Linux 2.6 后的内核特点 | 56 | 4.9 模块的编译 | 101 |
| 3.3 Linux 内核的组成 | 59 | 4.10 使用模块“绕开”GPL | 102 |
| 3.3.1 Linux 内核源代码的目录 结构 | 59 | 4.11 总结 | 103 |
| 3.3.2 Linux 内核的组成部分 | 60 | 第5章 Linux 文件系统与设备 文件 | 104 |
| 3.3.3 Linux 内核空间与用户 空间 | 64 | 5.1 Linux 文件操作 | 104 |
| 3.4 Linux 内核的编译及加载 | 64 | 5.1.1 文件操作系统调用 | 104 |
| 3.4.1 Linux 内核的编译 | 64 | 5.1.2 C 库文件操作 | 108 |
| 3.4.2 Kconfig 和 Makefile | 66 | 5.2 Linux 文件系统 | 109 |
| 3.4.3 Linux 内核的引导 | 74 | 5.2.1 Linux 文件系统目录结构 | 109 |
| 3.5 Linux 下的 C 编程特点 | 75 | 5.2.2 Linux 文件系统与设备 驱动 | 110 |
| 3.5.1 Linux 编码风格 | 75 | 5.3 devfs | 114 |
| 3.5.2 GNU C 与 ANSI C | 78 | 5.4 udev 用户空间设备管理 | 116 |
| 3.5.3 do { } while(0) 语句 | 83 | 5.4.1 udev 与 devfs 的区别 | 116 |
| 3.5.4 goto 语句 | 85 | 5.4.2 sysfs 文件系统与 Linux 设备 模型 | 119 |
| 3.6 工具链 | 85 | 5.4.3 udev 的组成 | 128 |
| 3.7 实验室建设 | 88 | 5.4.4 udev 规则文件 | 129 |
| 3.8 串口工具 | 89 | 5.5 总结 | 133 |
| 3.9 总结 | 91 | 第6章 字符设备驱动 | 134 |
| 第4章 Linux 内核模块 | 92 | 6.1 Linux 字符设备驱动结构 | 134 |
| 4.1 Linux 内核模块简介 | 92 | 6.1.1 cdev 结构体 | 134 |
| 4.2 Linux 内核模块程序结构 | 95 | 6.1.2 分配和释放设备号 | 136 |
| 4.3 模块加载函数 | 95 | 6.1.3 file_operations 结构体 | 136 |
| 4.4 模块卸载函数 | 97 | 6.1.4 Linux 字符设备驱动的 组成 | 138 |
| 4.5 模块参数 | 97 | 6.2 globalmem 虚拟设备实例描述 | 142 |

| | | | | | |
|------------------------------------|------------------------------|------------------------------|---|----------------------------------|-----|
| 6.3 | globalmem 设备驱动 | 142 | 第 8 章 Linux 设备驱动中的阻塞 与非阻塞 I/O | 189 | |
| 6.3.1 | 头文件、宏及设备结构体 | 142 | 8.1 阻塞与非阻塞 I/O | 189 | |
| 6.3.2 | 加载与卸载设备驱动 | 143 | 8.1.1 等待队列 | 191 | |
| 6.3.3 | 读写函数 | 144 | 8.1.2 支持阻塞操作的 globalfifo 设备驱动 | 194 | |
| 6.3.4 | seek 函数 | 146 | 8.1.3 在用户空间验证 globalfifo 的读写 | 198 | |
| 6.3.5 | ioctl 函数 | 146 | 8.2 轮询操作 | 198 | |
| 6.3.6 | 使用文件私有数据 | 148 | 8.2.1 轮询的概念与作用 | 198 | |
| 6.4 | globalmem 驱动在用户空间中的 验证 | 156 | 8.2.2 应用程序中的轮询编程 | 199 | |
| 6.5 | 总结 | 157 | 8.2.3 设备驱动中的轮询编程 | 201 | |
| 第 7 章 Linux 设备驱动中的并发 控制 | 158 | 8.3 支持轮询操作的 globalfifo 驱动 | 202 | 8.3.1 在 globalfifo 驱动中增加 轮询操作 | 202 |
| 7.1 | 并发与竞态 | 158 | 8.3.2 在用户空间中验证 globalfifo 设备的轮询 | 203 | |
| 7.2 | 编译乱序和执行乱序 | 160 | 8.4 总结 | 205 | |
| 7.3 | 中断屏蔽 | 165 | 第 9 章 Linux 设备驱动中的异步 通知与异步 I/O | 206 | |
| 7.4 | 原子操作 | 166 | 9.1 异步通知的概念与作用 | 206 | |
| 7.4.1 | 整型原子操作 | 167 | 9.2 Linux 异步通知编程 | 207 | |
| 7.4.2 | 位原子操作 | 168 | 9.2.1 Linux 信号 | 207 | |
| 7.5 | 自旋锁 | 169 | 9.2.2 信号的接收 | 208 | |
| 7.5.1 | 自旋锁的使用 | 169 | 9.2.3 信号的释放 | 210 | |
| 7.5.2 | 读写自旋锁 | 173 | 9.3 支持异步通知的 globalfifo 驱动 | 212 | |
| 7.5.3 | 顺序锁 | 174 | 9.3.1 在 globalfifo 驱动中增加异步 通知 | 212 | |
| 7.5.4 | 读-复制-更新 | 176 | 9.3.2 在用户空间中验证 globalfifo 的异步通知 | 214 | |
| 7.6 | 信号量 | 181 | | | |
| 7.7 | 互斥体 | 183 | | | |
| 7.8 | 完成量 | 184 | | | |
| 7.9 | 增加并发控制后的 globalmem 的 设备驱动 | 185 | | | |
| 7.10 | 总结 | 188 | | | |

| | | | | | |
|--------------------------|--------------------------|------------|---------------------------------|-----------------------------|-----|
| 9.4 | Linux 异步 I/O | 215 | 11.3.2 | 内核空间内存动态申请 | 262 |
| 9.4.1 | AIO 概念与 GNU C 库 AIO | 215 | 11.4 | 设备 I/O 端口和 I/O 内存的访问 | 267 |
| 9.4.2 | Linux 内核 AIO 与 libaio | 219 | 11.4.1 | Linux I/O 端口和 I/O 内存访问接口 | 267 |
| 9.4.3 | AIO 与设备驱动 | 222 | 11.4.2 | 申请与释放设备的 I/O 端口和 I/O 内存 | 268 |
| 9.5 | 总结 | 223 | 11.4.3 | 设备 I/O 端口和 I/O 内存访问流程 | 269 |
| 第 10 章 中断与时钟 | | 224 | 11.4.4 | 将设备地址映射到用户空间 | 270 |
| 10.1 | 中断与定时器 | 224 | 11.5 | I/O 内存静态映射 | 276 |
| 10.2 | Linux 中断处理程序架构 | 227 | 11.6 | DMA | 277 |
| 10.3 | Linux 中断编程 | 228 | 11.6.1 | DMA 与 Cache 一致性 | 278 |
| 10.3.1 | 申请和释放中断 | 228 | 11.6.2 | Linux 下的 DMA 编程 | 279 |
| 10.3.2 | 使能和屏蔽中断 | 230 | 11.7 | 总结 | 285 |
| 10.3.3 | 底半部机制 | 230 | 第 12 章 Linux 设备驱动的软件架构思想 | | |
| 10.3.4 | 实例: GPIO 按键的中断 | 235 | 286 | | |
| 10.4 | 中断共享 | 237 | 12.1 | Linux 驱动的软件架构 | 286 |
| 10.5 | 内核定时器 | 238 | 12.2 | platform 设备驱动 | 290 |
| 10.5.1 | 内核定时器编程 | 238 | 12.2.1 | platform 总线、设备与驱动 | 290 |
| 10.5.2 | 内核中延迟的工作 delayed_work | 242 | 12.2.2 | 将 globalfifo 作为 platform 设备 | 293 |
| 10.5.3 | 实例: 秒字符设备 | 243 | 12.2.3 | platform 设备资源和数据 | 295 |
| 10.6 | 内核延时 | 247 | 12.3 | 设备驱动的分层思想 | 299 |
| 10.6.1 | 短延迟 | 247 | 12.3.1 | 设备驱动核心层和例化 | 299 |
| 10.6.2 | 长延迟 | 248 | 12.3.2 | 输入设备驱动 | 301 |
| 10.6.3 | 睡着延迟 | 248 | 12.3.3 | RTC 设备驱动 | 306 |
| 10.7 | 总结 | 250 | 12.3.4 | Framebuffer 设备驱动 | 309 |
| 第 11 章 内存与 I/O 访问 | | 251 | 12.3.5 | 终端设备驱动 | 311 |
| 11.1 | CPU 与内存、I/O | 251 | 12.3.6 | misc 设备驱动 | 316 |
| 11.1.1 | 内存空间与 I/O 空间 | 251 | | | |
| 11.1.2 | 内存管理单元 | 252 | | | |
| 11.2 | Linux 内存管理 | 256 | | | |
| 11.3 | 内存存取 | 261 | | | |
| 11.3.1 | 用户空间内存动态申请 | 261 | | | |

| | | | |
|--|------------|---|------------|
| 12.3.7 驱动核心层 | 321 | 第 14 章 Linux 网络设备驱动 | 358 |
| 12.4 主机驱动与外设驱动分离的设计思想 | 321 | 14.1 Linux 网络设备驱动的结构 | 358 |
| 12.4.1 主机驱动与外设驱动分离 | 321 | 14.1.1 网络协议接口层 | 359 |
| 12.4.2 Linux SPI 主机和设备驱动 | 322 | 14.1.2 网络设备接口层 | 363 |
| 12.5 总结 | 330 | 14.1.3 设备驱动功能层 | 367 |
| 第 13 章 Linux 块设备驱动 | 331 | 14.2 网络设备驱动的注册与注销 | 367 |
| 13.1 块设备的 I/O 操作特点 | 331 | 14.3 网络设备的初始化 | 369 |
| 13.2 Linux 块设备驱动结构 | 332 | 14.4 网络设备的打开与释放 | 370 |
| 13.2.1 block_device_operations 结构体 | 332 | 14.5 数据发送流程 | 371 |
| 13.2.2 gendisk 结构体 | 334 | 14.6 数据接收流程 | 372 |
| 13.2.3 bio、request 和 request_queue | 335 | 14.7 网络连接状态 | 375 |
| 13.2.4 I/O 调度器 | 339 | 14.8 参数设置和统计数据 | 377 |
| 13.3 Linux 块设备驱动的初始化 | 340 | 14.9 DM9000 网卡设备驱动实例 | 380 |
| 13.4 块设备的打开与释放 | 342 | 14.9.1 DM9000 网卡硬件描述 | 380 |
| 13.5 块设备驱动的 ioctl 函数 | 342 | 14.9.2 DM9000 网卡驱动设计分析 | 380 |
| 13.6 块设备驱动的 I/O 请求处理 | 343 | 14.10 总结 | 386 |
| 13.6.1 使用请求队列 | 343 | 第 15 章 Linux I²C 核心、总线与设备驱动 | 387 |
| 13.6.2 不使用请求队列 | 347 | 15.1 Linux I ² C 体系结构 | 387 |
| 13.7 实例: vmem_disk 驱动 | 349 | 15.2 Linux I ² C 核心 | 394 |
| 13.7.1 vmem_disk 的硬件原理 | 349 | 15.3 Linux I ² C 适配器驱动 | 396 |
| 13.7.2 vmem_disk 驱动模块的加载与卸载 | 349 | 15.3.1 I ² C 适配器驱动的注册与注销 | 396 |
| 13.7.3 vmem_disk 设备驱动的 block_device_operations | 351 | 15.3.2 I ² C 总线的通信方法 | 397 |
| 13.7.4 vmem_disk 的 I/O 请求处理 | 352 | 15.4 Linux I ² C 设备驱动 | 399 |
| 13.8 Linux MMC 子系统 | 354 | 15.4.1 Linux I ² C 设备驱动的模块加载与卸载 | 400 |
| 13.9 总结 | 357 | 15.4.2 Linux I ² C 设备驱动的数据传输 | 400 |

| | | | |
|--|------------|--|------------|
| 15.4.3 Linux 的 i2c-dev.c 文件 分析 | 400 | 16.5 USB OTG 驱动 | 456 |
| 15.5 Tegra I ² C 总线驱动实例 | 405 | 16.6 总结 | 458 |
| 15.6 AT24xx EEPROM 的 I ² C 设备 驱动实例 | 410 | 第 17 章 I²C、SPI、USB 驱动 架构类比 | 459 |
| 15.7 总结 | 413 | 17.1 I ² C、SPI、USB 驱动架构 | 459 |
| 第 16 章 USB 主机、设备与 Gadget 驱动 | 414 | 17.2 I ² C 主机和外设眼里的 Linux 世界 | 460 |
| 16.1 Linux USB 驱动层次 | 414 | 第 18 章 ARM Linux 设备树 | 461 |
| 16.1.1 主机侧与设备侧 USB 驱动 | 414 | 18.1 ARM 设备树起源 | 461 |
| 16.1.2 设备、配置、接口、 端点 | 415 | 18.2 设备树的组成和结构 | 462 |
| 16.2 USB 主机控制器驱动 | 420 | 18.2.1 DTS、DTC 和 DTB 等 | 462 |
| 16.2.1 USB 主机控制器驱动的 整体结构 | 420 | 18.2.2 根节点兼容性 | 468 |
| 16.2.2 实例：Chipidea USB 主机 驱动 | 425 | 18.2.3 设备节点兼容性 | 470 |
| 16.3 USB 设备驱动 | 425 | 18.2.4 设备节点及 label 的命名 | 475 |
| 16.3.1 USB 设备驱动的整体 结构 | 425 | 18.2.5 地址编码 | 477 |
| 16.3.2 USB 请求块 | 430 | 18.2.6 中断连接 | 479 |
| 16.3.3 探测和断开函数 | 435 | 18.2.7 GPIO、时钟、pinmux 连接 | 480 |
| 16.3.4 USB 骨架程序 | 436 | 18.3 由设备树引发的 BSP 和驱动 变更 | 484 |
| 16.3.5 实例：USB 键盘驱动 | 443 | 18.4 常用的 OF API | 490 |
| 16.4 USB UDC 与 Gadget 驱动 | 446 | 18.5 总结 | 493 |
| 16.4.1 UDC 和 Gadget 驱动的 关键数据结构与 API | 446 | 第 19 章 Linux 电源管理的系统 架构和驱动 | 494 |
| 16.4.2 实例：Chipidea USB UDC 驱动 | 451 | 19.1 Linux 电源管理的全局架构 | 494 |
| 16.4.3 实例：Loopback Function 驱动 | 453 | 19.2 CPUFreq 驱动 | 495 |
| | | 19.2.1 SoC 的 CPUFreq 驱动 实现 | 495 |
| | | 19.2.2 CPUFreq 的策略 | 501 |

| | | | |
|----------------------------------|------------|------------------------------|------------|
| 19.2.3 CPUFreq 的性能测试和调优 | 501 | 20.6 GPIO 驱动 | 557 |
| 19.2.4 CPUFreq 通知 | 502 | 20.7 pinctrl 驱动 | 560 |
| 19.3 CPUIdle 驱动 | 504 | 20.8 时钟驱动 | 572 |
| 19.4 PowerTop | 508 | 20.9 dmaengine 驱动 | 578 |
| 19.5 Regulator 驱动 | 508 | 20.10 总结 | 580 |
| 19.6 OPP | 511 | 第 21 章 Linux 设备驱动的调试 | 581 |
| 19.7 PM QoS | 515 | 21.1 GDB 调试器的用法 | 581 |
| 19.8 CPU 热插拔 | 518 | 21.1.1 GDB 的基本用法 | 581 |
| 19.9 挂起到 RAM | 522 | 21.1.2 DDD 图形界面调试工具 | 591 |
| 19.10 运行时的 PM | 528 | 21.2 Linux 内核调试 | 594 |
| 19.11 总结 | 534 | 21.3 内核打印信息——printk() | 596 |
| 第 20 章 Linux 芯片级移植及底层驱动 | 535 | 21.4 DEBUG_LL 和 EARLY_PRINTK | 599 |
| 20.1 ARM Linux 底层驱动的组合和现状 | 535 | 21.5 使用“/proc” | 600 |
| 20.2 内核节拍驱动 | 536 | 21.6 Oops | 606 |
| 20.3 中断控制器驱动 | 541 | 21.7 BUG_ON() 和 WARN_ON() | 608 |
| 20.4 SMP 多核启动以及 CPU 热插拔驱动 | 549 | 21.8 strace | 609 |
| 20.5 DEBUG_LL 和 EARLY_PRINTK 的设置 | 556 | 21.9 KGDB | 610 |
| | | 21.10 使用仿真器调试内核 | 612 |
| | | 21.11 应用程序调试 | 613 |
| | | 21.12 Linux 性能监控与调优工具 | 616 |
| | | 21.13 总结 | 618 |