Broadview
www.broadview.com.cn
华清远见 FARSIGHT 始于2004

高等院校嵌入式人才培养规划教材

嵌入式Linux驱动开发教程

/ 注重实践性、实用性，重点突出 /
/ 知识体系清晰、完整，内容逻辑性强，实验案例丰富，适合系统学习 /
/ 所有实验均可在基于ARM Cortex-A9的FS4412平台上验证 /

华清远见嵌入式学院
姜先刚 刘洪涛 编著

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

院校嵌入式人才培养规划教材

嵌入式Linux驱动开发教程

华清远见嵌入式学院
姜先刚 刘洪涛 编著

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

内 容 简 介

本书结合大量实例，在基于 ARM Cortex-A9 四核处理器 Exynos4412 的硬件教学平台和 PC 上，全面详细地讲解了 Linux 设备驱动开发。主要内容包括开发环境的搭建、内核模块、字符设备驱动框架、高级 I/O、中断和时间管理、互斥和同步、内存和 DMA、Linux 设备模型、外设的驱动实例、总线类设备驱动、块设备驱动、网络设备驱动和内核调试技术。每一个知识点都有一个对应的典型实例，大多数实例既可以在上面说到的嵌入式平台上运行，也可以在 PC 上运行。另外，本书也引入了新内核的一些新特性，比如高分辨率定时器、针对嵌入式平台的 dmaengine 和设备树。在需要重点关注的地方还加入了大量的内核源码分析，使读者能够快速并深刻理解 Linux 设备驱动的开发。

本书可作为大学院校电子、通信、计算机、自动化等专业的嵌入式 Linux 设备驱动开发课程的教材，也可供嵌入式 Linux 驱动开发人员参考。

图书在版编目（CIP）数据

嵌入式 Linux 驱动开发教程 / 华清远见嵌入式学院，姜先刚，刘洪涛编著. —北京：电子工业出版社，2017.6

高等院校嵌入式人才培养规划教材

ISBN 978-7-121-31359-2

Ⅰ. ①嵌… Ⅱ. ①华… ②姜… ③刘… Ⅲ. ①Linux 操作系统—程序设计—高等学校—教材 Ⅳ.①TP316.89

中国版本图书馆 CIP 数据核字（2017）第 077637 号

策划编辑：孙学瑛
责任编辑：徐津平
特约编辑：赵树刚
印　　刷：北京京科印刷有限公司
装　　订：北京京科印刷有限公司
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：787×1092　1/16　印张：25　字数：640 千字
版　　次：2017 年 6 月第 1 版
印　　次：2017 年 6 月第 1 次印刷
定　　价：69.00 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到 dbqq@phei.com.cn。

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

前　言

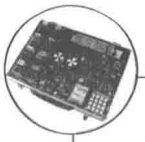
随着嵌入式及物联网技术地快速发展，ARM 处理器已经广泛地应用到了工业控制、智能仪表、汽车电子、医疗电子、军工电子、网络设备、消费类电子、智能终端等领域。而较新的 ARM Cortex-A9 架构的四核处理器更是由于其优越的性能被广泛应用到了中高端的电子产品市场。比如基于 ARM Cortex-A9 的三星 Exynos4412 处理器就被应用在了三星 GALAXY Note II 智能手机上。

另外，Linux 内核由于其高度的稳定性和可裁剪性等特点，被广泛地应用到了嵌入式系统，Android 系统就是一个典型的例子。这样，ARM 处理器就和 Linux 操作系统紧密地联系在了一起。所以，基于 ARM 和 Linux 的嵌入式系统就得到了快速的发展。

嵌入式系统是一个定制的系统，所以千变万化、形形色色的硬件都必须要有对应的驱动才能使其正常工作，为这些硬件设备编写驱动就是不可避免的了。虽然有很多内核开发人员已经为很多常见的硬件开发了驱动，但是驱动的升级一般都跟不上新硬件的升级。笔者就多次遇到过内核的驱动和同一系列的升级版本芯片不匹配的情况，这时就要改写驱动程序。所以内核层次的底层开发几乎都要和驱动打交道。另外，了解驱动（或者说内核）的一些底层工作原理，也有助于我们写出更稳定、更高效的应用层代码。

为了能够实现这一目标，并促进嵌入式技术的推广，华清远见研发中心自主研发了一套基于 Exynos4412 处理器的开发板 FS4412，并组织编写了本书。本书注重实践、实用，没有用长篇大论来反复强调一些旁枝末节的内容，但是对于会影响理解的部分又非常详细地分析了内核源码，并给出了大量的图示。书中的各个实例虽然为了突出相关的知识重点而简化了对某些问题的讨论，不能称得上工程上严格意义的好驱动，但是确实也具备了对应的设备驱动开发所必需的各方面。实例按照工程上驱动开发的增量式方式来进行，即先有主体再逐渐完善，循序渐进。读者按照实例能够迅速掌握对应驱动的开发精要，对整个驱动的实现也就有了一个清晰的思路。

本书共 14 章，循序渐进地讲解了嵌入式 Linux 设备驱动开发所涉及的理论基础和大量 API 说明，并配有大量驱动实例。全书主要分为五部分：第一部分是 Linux 设备驱动开发的概述，包含第 1 章；第二部分是模块及字符设备驱动的理论，包含第 2～8 章；第三部分是字符设备驱动实例，包含第 9 章和第 10 章；第四部分是 Linux 块设备驱动和网络设备驱动，包含第 11 章和第 12 章；最后一部分是 Linux 内核的调试和开发环境的搭



建，包含第 13 章和第 14 章。各章节的主要内容如下。

第 1 章概述了需要了解 Linux 驱动程序的人群、Linux 驱动开发的特点和本书其他各章节的核心内容。

第 2 章对 Linux 内核的模块进行了介绍，现在的驱动几乎都以 Linux 内核模块的形式来实现，所以这是后续的基础。

第 3 章讲解 Linux 字符设备驱动的主体框架，并以一个假想的串口来实现驱动。这是 Linux 设备驱动入门的关键，所以分析了大量的内核源码。当然，这个驱动是不完善的，需要在后面的各章节逐步添加功能。

第 4 章在上一章的基础上探讨了字符设备的高级 I/O 操作，包括 ioctl、阻塞、I/O 多路复用、异步通知、mmap、定位等，还特别介绍了 proc 相关的接口。

第 5 章讲解中断和时间管理，为便于理解，特别加入了中断进入的内核源码分析。时间管理则包含了延时和定时两部分，在定时部分还讨论了新内核中的高分辨率定时器。

第 6 章讲解了互斥和同步，为了让读者明白互斥对驱动开发的重要性，特别从 ARM 汇编的层次来讨论了竞态。除了对传统的互斥（自旋锁、信号量等）进行讨论外，还特别说明了 RCU 机制和使用的范例。

第 7 章讲解了内核中内存的各种分配方式，还特别谈到了 per-CPU 变量的使用。最后，对 DMA 的讨论则专注于新内核引入的 dmaengine 子系统，并用一个实例进行了具体的展现。

第 8 章讲解了 Linux 设备模型，这部分内容比较抽象。为了能帮助读者理解这部分内容，专门实现了设备、总线、驱动三个最简单的实例，从而使读者完全掌握三者之间的关系。这一章的后半部分有大量实用技术的展现，包括电源管理、驱动的自动加载、设备节点的自动创建等。最后还讲解了较新的内核引入的 ARM 体系结构的设备树。

第 9 章在前面的理论基础上实现了大量外设的驱动。这些驱动并不都是通过字符设备框架来实现的，目的就是想告诉读者，如果我们能够简化驱动的编写，就尽量简化驱动的编写，多使用内核中已经实现的机制。

第 10 章讲解了总线类设备驱动的开发，对流行的 I2C 总线、SPI 总线、USB 总线和 PCI 总线都进行了讨论。这些总线都有一个共同的特性，就是都有主机控制器和连接在总线上的设备，我们只讨论了在主机控制器驱动之上的设备驱动，不讨论主机控制器驱动及设备自身的固件或驱动，因为设备驱动是最常开发的驱动。

第 11 章讲解了块设备驱动，为了便于读者对这部分知识进行理解，特别介绍了磁盘的内部结构，然后用内存虚拟了一个磁盘，用两种方式实现了该虚拟磁盘的块设备驱动。

第 12 章讲解了网络设备驱动，用一个虚拟的环回以太网卡的驱动展现了网络设备驱动的主体框架，还分析了 DM9000 网卡驱动的主体框架部分，并和前面的虚拟网卡驱动进行了对比。

第 13 章介绍了内核的一些调试技术。内核的调试相对来说比较麻烦，但只要能熟练使用这些调试技术，还是能较快找出问题所在的。

第 14 章是嵌入式 Linux 设备驱动开发环境的搭建，包含了主机系统的准备和各个软件的安装。尤其是用 vim 搭建了一个适合于驱动开发的类似于 IDE 的编辑环境，能够大大提高代码的编写效率。

本书由华清远见成都中心的姜先刚编写，北京中心的刘洪涛承担全书的统稿及审校工作，是贾燕枫、杨曼、袁祖刚、关晓强、谭翠君、李媛媛、张丹、张志华、曹忠明、苗德行、冯利美、卢闫进、蔡蒙等老师心血的结晶，也是他们多年教学成果的积累。他们认真阅读了书稿，提出了大量的建议，并纠正了书稿中的很多错误，在此特表示感谢。

由于作者水平有限，书中不妥之处在所难免，恳请读者批评指正。对于本书的批评和建议，可以发表到 www.farsight.com.cn 技术论坛。

编 者

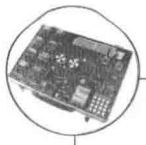
2017 年 3 月

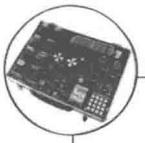
轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **提交勘误**：您对书中内容的修改意见可在【提交勘误】处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **与作者交流**：在页面下方【读者评论】处留下您的疑问或观点，与作者和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/31359

目 录

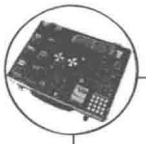




第1章 概述

本章目标

本章首先概要性地介绍了需要了解 Linux 设备驱动程序的人群；然后在不涉及过多具体知识点的情况下讨论了 Linux 驱动程序的部分显著特点；最后概要说明了本书各个章节的核心内容。



Linux 是一款成功的、优秀的开源项目，随着应用的日益广泛，Linux 已受到越来越多的软件开发者的追捧。但是从官网上下载了源码并解压后，我们往往会迷失在浩瀚的代码海洋中，这巨大的代码量令很多人望而却步。那么，我们是不是就没有办法征服它了呢？记得很有经验的前辈曾提到过以下两个突破口：驱动和网络。确实，如果对 Linux 的各部分内核源码做一个统计，会发现这两部分代码所占的比例是最高的。本书是一本关于驱动开发的入门级教程，希望它能帮助各位 Linux 内核初学者找到学习的突破口，打开 Linux 内核的大门，能够掌握 Linux 驱动程序的开发。

总体来说，专门从事 Linux 驱动开发的工程师并不是特别多，但是这并不是说我们就完全没有和它打交道。笔者认为，除了专职的 Linux 驱动开发工程师之外，学习或了解 Linux 驱动开发对下面几类开发人员也是有益的。

（1）Linux 系统移植工程师。虽然现在 ARM 体系结构的 Linux 内核也引入了设备树，使内核移植工程师几乎不需要再改写 BSP 文件，只是修改设备树和对内核进行选配就可以了，但是移植的过程并不总是特别顺利。笔者在移植 LCD 驱动时就发现，官网源码中提供的设备树节点在驱动中不能被正确识别，要修改节点的编写形式。也就是说，要能写好一个设备树节点，除了参考内核文档，有时还要参考驱动源码，所以能看懂相应的驱动源码很重要。另外，所选配的驱动有时不能使硬件完美工作起来，还是以前面的 LCD 驱动为例，笔者发现驱动中假设了 U-Boot 已经将 LCD 的时钟初始化好了，但事实情况是 U-Boot 并没有初始化这部分时钟。为了使驱动的通用性更强，笔者决定在驱动中添加相关的时钟初始化代码。这就说明 Linux 系统移植工程师几乎必须要和驱动打交道。而且，Linux 驱动的更新不一定能跟得上硬件的更新，当使用了同一系列中较新的芯片时，往往也会涉及驱动代码的修改。

（2）内核应用开发工程师。这其实就是内核开发工程师，只是他们主要调用 Linux 内核的 API 接口来完成某些特定的功能。比如，开一个内核线程来完成某件事情，绕过文件系统相关的代码直接访问磁盘，过滤一些网络数据等。这类开发者直接在内核层工作，通常以优化性能、提高程序效率或进行底层的监控为目标。这些代码都很有可能会调用驱动所提供的接口。

（3）系统编程工程师。主要调用系统调用接口来完成应用软件的编程，通常是一些平台软件，如 SDK，为更上层的应用开发者提供编程接口。了解底层的工作原理有助于写出更稳定、更高效的应用程序。

Linux 内核源码是由全世界的众多优秀开发者共同开发出来的，它的稳定性足以证明其设计的合理性，从事软件开发的人员应该都能够从中得到一些启发或有所借鉴。

那么，Linux 驱动开发和一般的应用开发有哪些区别，或者说，这种开发具有哪些最鲜明的特点呢？

（1）Linux 驱动开发是内核级别的开发，驱动程序的任何问题都可能引起整个系统的崩溃。因为应用程序由操作系统来管理，应用程序的崩溃通常不会影响到其他的程序或整个系统，造成的破坏是比较小的。比如，应用程序对非法指针进行解引用，通常只会

引发一个段错误，然后程序本身崩溃而已。但是如果驱动对一个非法指针进行解引用，通常就会导致整个内核的崩溃。

（2）驱动程序通常都要进行中断处理，在中断上下文（或类似的原子上下文）中的编程有比较严格的限制（这在相关的章节会做更详细的描述），处理不好也会导致内核崩溃。而在应用程序中则没有这些方面的内容。

（3）驱动程序有更多的并发环境需要考虑，比如上面说到的中断，以及在多处理器系统下的驱动编程。一个好的驱动，不应该假设自己的运行环境，或者说，都应该假设运行在各种并发环境下。

（4）驱动程序是被动接受上层调用的代码，是为上层提供服务的一套代码，所以我们会在驱动中看到很多注册和注销的函数。

（5）一类驱动程序都有一个特定的实现模板，在这里姑且称之为驱动的框架。另外，所有的驱动都有一种类似的实现模式，就是构造核心的数据结构，然后注册到内核。学习驱动开发，主要是学习这些核心的数据结构和与之相关的一套 API。编写驱动则是按照特定类型驱动的框架来构造这些核心数据结构，然后再注册到内核。驱动中灵活的部分是这些框架规定的接口函数的实现。

（6）驱动程序虽然是用 C 语言来开发的，但是很多地方都体现了面向对象的编程思想，这在 Linux 设备模型中体现得尤为突出。

（7）应该尽量利用内核中已有的实现，而不是自己重新构建，本书中有几个驱动的例子都是这样的。而且，内核源码中有同一类设备驱动很好的实现范例，面对不熟悉的驱动框架时，可以参考内核源码，从而快速掌握该类驱动的开发。

（8）Linux 内核是基于 GNU C 进行开发的，它对标准的 C 语言做了一些扩展，但是本书的示例代码都避免使用这些扩展的特性，尽量和标准 C 语言一致。而且，Linux 内核代码有一套编码风格，大家可以参考 https://www.kernel.org/doc/Documentation/CodingStyle，在此就不细说了。

上面的内容只是概要性的描述，具体的知识点在本书的具体章节会进行进一步地归纳和总结。接下来对本书各个章节和核心内容做一个简要的说明。

Linux 的内核源码编译后将会生成一个总的镜像，将该镜像加载到内存中并运行之，就会启动内核。驱动属于内核代码的一部分，每次对驱动做任何修改都会重新编译整个内核，还要重新加载运行内核，这个时间消耗是比较大的。所以如果驱动能独立于内核镜像之外，并能动态加载和卸载，那么驱动开发的时间消耗将会大大降低。本书第 2 章讨论的就是这个独立于内核之外，并能动态加载和卸载的模块。

有一类设备的数据访问是按字节流的方式来进行的，也就是访问的单位可以小到字节，比如键盘、鼠标等。针对这一类设备的 Linux 驱动叫字符设备驱动，是开发中最有可能遇到的一类驱动，也是内核中最多的一类驱动。我们前面说过，学习一类驱动就是要学习它的核心数据结构和一组 API，然后是由此组成的框架。我们在第 3 章将会首次接触到这个概念，也会注册一些接口函数，从而为上层提供服务，或者说供上层调用。

掌握了这些概念后，应用同样的方式，就可以快速地掌握其他驱动的框架。

在类 UNIX 系统中，设备也被当成文件来对待，或者说将设备抽象成了文件。这样就可以统一应用层代码对普通文件和设备文件的访问接口。对于文件的操作，有很多种 I/O 模型，比如大家非常熟悉的阻塞、非阻塞，或者 Windows 系统经常提到的同步和异步。既然设备被抽象成了文件，而设备又是最终被驱动程序所管理的，自然设备的驱动就应该提供这些 I/O 模型的支持，本书第 4 章将重点阐述这方面的内容。

前面说过，Linux 驱动编程相对于应用程序编程多了中断的处理。因为驱动是管理硬件的，而为了提高硬件的访问效率，通常不是由 CPU 来轮询硬件的状态，而是在硬件准备好后主动通知 CPU，这种硬件上的异步通知就是中断。第 5 章将会讨论在驱动中如何编写中断服务例程，以及中断服务例程中的限制和一些应对策略。另外，传统的硬件定时器也是以中断方式工作的，所以在这一章还讨论了延时和定时方面的内容。

前面也提到过，在内核中的并发方式多于应用层中遇到的并发。为了避免并发的执行路径对共享数据访问带来的相互覆盖的问题，我们需要认真处理好这个问题。不管在应用层还是在内核层，数据的紊乱都可能导致灾难性的结果，但是因为内核层出现这种情况的时候更多，关系更复杂，所以要更谨慎对待之。当然，内核提供的解决方案也要更多一些，不过这也带来了另一个问题，就是如何从这些方案中选择一种最合适的方案。本书第 6 章将会详细讨论这方面的内容。

关于内存的使用，内核层提供了更多的选择。Linux 向来以高效、稳定著称，内存是计算机系统中的宝贵资源，在内存的管理上下再多的功夫都不为过。当然，Linux 在这方面考虑得很周全，所以它才能在小到手表大小的嵌入式系统上和大到集群服务器这样的系统上都能运行自如。在本书第 7 章我们将会从驱动编程实用性的角度来学习相关的知识点。另外，驱动会利用 DMA 操作来减轻 CPU 的负担，所以第 7 章也会讨论在嵌入式系统上的 DMA 编程接口。

自从 Linus 抱怨了 ARM 体系结构相关内核代码的混乱后，ARM 社区就开始积极处理这个问题，最后引入了设备树。在第 8 章中，我们首先会细数前面讲解的驱动开发模式的弊端，然后逐渐按照历史的顺序来还原这个 Linux 设备模型的产生和升级过程。在第 8 章也会详细阐明驱动开发中的设备和驱动分离的思想，这种思想使驱动能够动态获取设备的信息，而不是将设备信息硬编码在驱动中，从而提高了驱动的灵活性。这也是减轻 Linux 系统移植工作量的关键所在。

学习驱动的目的最终是为了能为各种各样的设备写出驱动代码。本书第 9 章将会针对一个嵌入式目标板上常见的外设，从原理图和芯片手册出发，配合驱动的框架，来逐一实现这些设备的驱动。而且，我们要善于利用内核中已有的设施，以最快、最简单的方式来实现设备的驱动。所以在第 9 章，部分外设在自己用代码实现了驱动后，还会分析内核中已有的驱动，并利用这些驱动来达到访问、控制设备的目的。还有一类连接在总线上的设备，在第 10 章将会对之进行讨论。第 10 章首先分析了这些总线类设备驱动的框架，然后再用实例来做展示。

Linux 除了前面谈到的字符设备驱动这一大类，还有块设备驱动和网络设备驱动这两大类。相对于字符设备驱动而言，这两类设备的驱动要少一些，但是却更复杂一些。这包含两个方面，第一是框架要复杂一些，涉及的内核组件要多一些；第二是硬件本身要复杂一些。为了突出框架的主体，避免过多涉及硬件的细节，我们选择用虚拟硬件的方式来实现这两类设备的驱动。

出于完整性考虑，本书的第 13 章讨论了内核的几种调试方法。第 14 章详细描述了 Linux 驱动程序开发环境的搭建，如果对 Linux 驱动开发环境不熟悉的读者，请先阅读第 14 章的内容。

第2章 内核模块

本章目标

绝大多数的驱动都是以内核模块的形式来实现的。本章主要围绕什么是内核模块，以及如何编写、编译、加载并测试模块程序来展开。另外，本章还将讨论模块的一些其他重要特性。

- 第一个内核模块程序
- 内核模块的相关工具
- 内核模块一般的形式
- 将多个源文件编译生成一个内核模块
- 内核模块参数
- 内核模块依赖
- 关于内核模块的进一步讨论