Broadview
www.broadview.com.cn

深入 Linux 设备驱动程序内核机制

Internals of Linux Device Driver

◎ 陈学松 著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

深入Linux设备驱动程序内核机制

Internals of Linux Device Driver

◎ 陈学松 著

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

内 容 简 介

这是一本系统阐述 Linux 设备驱动程序技术内幕的专业书籍，它的侧重点不是讨论如何在 Linux 系统下编写设备驱动程序，而是要告诉读者隐藏在这些设备驱动程序背后的那些内核机制及原理。作者通过对 Linux 内核源码抽丝剥茧般的解读，再辅之以精心设计的大量图片，使读者在阅读完本书后对驱动程序前台所展现出来的那些行为特点变得豁然开朗。

本书涵盖了编写设备驱动程序所需要的几乎所有的内核设施，比如内核模块、中断处理、互斥与同步、内存分配、延迟操作、时间管理，以及新设备驱动模型等内容。为了避免读者迷失在某一技术细节的讨论当中，本书在一个比较高的层面上进行展开，以一种先框架再细节的结构安排极大地简化了读者的阅读与学习。

本书不仅适合那些在 Linux 系统下从事设备驱动程序开发的专业技术人员阅读，也同样适合有志于从事 Linux 设备驱动程序开发或对 Linux 设备驱动程序及 Linux 内核感兴趣的在校学生等阅读。对于没有任何 Linux 设备驱动程序开发经验的初学者，建议先阅读那些讨论“如何”在 Linux 系统下编写设备驱动程序的入门书籍，然后再阅读本书来理解“为什么”要以这样或者那样的方式来编写设备驱动程序。

图书在版编目（CIP）数据

深入 Linux 设备驱动程序内核机制 / 陈学松著. —北京：电子工业出版社，2012.1
ISBN 978-7-121-15052-4

Ⅰ. ①深… Ⅱ. ①陈… Ⅲ. ①Linux 操作系统－程序设计 Ⅳ. ①TP316.89

中国版本图书馆 CIP 数据核字（2011）第 231765 号

策划编辑：张春雨
责任编辑：白 涛
印 刷：
装 订：三河市鑫金马印装有限公司
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：33.75 字数：856 千字
印 次：2012 年 1 月第 1 次印刷
印 数：3000 册 定价：98.00 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。

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。

服务热线：（010）88258888。

推荐序

这不是一本单纯的关于 Linux 设备驱动程序入门的书。它是给有一定的 Linux 设备驱动程序编写经验并且对众多 Linux 底层设备驱动内幕机制感兴趣的读者量身定制的。与市面上已经出版的 Linux 相关方面的图书的不同之处在于，本书并不着重于全面描述 Linux 内核，也不只是简单地告诉你如何去写一个 Linux 下的设备驱动程序。它是从设备驱动程序的视角出发，深入到 Linux 内核去剖析那些和驱动程序实现机制密切相关的技术内幕。比如让你理解为什么在这个地方驱动程序应该使用 work queue 而不是 tasklet，为什么在中断处理例程里应该使用 spin_lock 而不是 mutex_lock……因为只有当你对驱动程序中使用的各种内核实现有了清晰的认识，你才能在日常的工作当中随心所欲地驾驭它们，写出更高性能更安全的代码。知其然，更知其所以然，对于沉迷于技术领域的人而言，这种不断探索的好奇心是对技术工作能长期保持热情的一个基本特质。相对于市面上已经出版的相关书籍而言，本书具有以下两个鲜明的特色：

细节揭秘

目前市场上已经出版的 Linux 内核和驱动程序方面的书籍，大体上可分为两种。一种是侧重于内核本身，鉴于目前 Linux 的内核源码已经十分庞大，这些讲解内核的书有些本身非常全面，作者的写作态度也非常严谨，比如 *Deep Understanding Linux Kernel*，还有新近出版的 *Professional Linux Kernel Architecture*，后者几乎涵盖了新版 Linux 内核中绝大部分重要的构件，但也正因如此，这样的书籍就不可能在与驱动程序相关的机制上留下太多笔墨。另外还有一种是专门讲解 Linux 驱动方面的书籍，典型的有 *Linux Device Driver* 和 *Essential Linux Device Driver*。这些书着重于介绍 Linux 驱动的基本概念和架构，但是对于想了解更多幕后的技术细节的读者来说，《深入 Linux 设备驱动程序内核机制》一书可提供更详细的资源和帮助。通常当你想深入理解一些一般书籍没有描述的机制时，你可能会采用在线搜索或查看源码的方式，但有时这不仅费时也未必能得到满意的答案。本书提供了另一途径让你更系统、有效地理解这些内核机制。我相信对于广大忙于在校学习、职场深造或课题攻关的读者来说，本书可提供很多有益的帮助。

图片说理

这本书另外一个很大的特点是，作者大量使用其精心设计的图片来帮助你清晰地理解一些复杂的概念、流程和架构。这在中文版原创的图书中是很难能可贵的，相对而言外文书在这方面做得就要好很多。形象直观的图片胜过大量的文字，也能节省读者大量的时间。可以看到，本书的作者在这一方面做了很大的努力去加以完善，在我看来，这是一个非常好的尝试。本书作者当前正在 AMD 上海研发中心从事 Linux 显卡驱动等系统软件方面的研发工作，能在繁忙的工作之余，通过对自己学习和实践经验的总结写下这样一本书，对增进国内读者的 Linux 系统开发能力将起到很大的作用。我相信，如果作者有足够的时间与精力的话，这本书还可以进一步完善，包括在某些技术方面可以有更精细的描述。

AMD 图形软件架构师　PMTS　俞辉

2011 年 8 月 24 日于加拿大

前　言

在 Linux 庞大的源码树中，设备驱动程序部分的代码已经占了相当大的比例。现实的工作中，大量的采用 Linux 系统的平台需要设备驱动程序才能把 Linux 的内核真正运行起来，同时通过编写 Linux 设备驱动程序，使得我们经由亲手编写具有特权等级的代码来一探 Linux 内核幕后的秘密成为可能。所以，无论是从日常工作的需要还是只为单纯满足对 Linux 内核机制好奇心的角度来说，学习并掌握 Linux 设备驱动程序的编写都是非常必要的，同时也是一件非常有趣且有意义的事情。

初衷与定位

这本书并不仅仅是单纯地讨论如何在 Linux 系统下编写一个设备驱动程序，因为关于这方面的内容，市面上已经有大量类似的图书可供参考。本书的总体思想是从内核的角度来看设备驱动程序，从设备驱动程序的角度深入到内核中，比如通过对 spin_lock 以及 spin_lock_irq 等内核源码的分析，来告诉你在什么场合下应该使用 spin_lock，什么场合下又应该选择 spin_lock_irq。还有，比如我们几乎每天都会在设备驱动程序所代表的内核模块中使用 MODULE_LICENSE("GPL")这样的声明，这个声明是如此地平凡，以至于我们常常忽略它存在的价值。但是在某个夜深人静的夜晚，感觉长夜漫漫无心睡眠时，在你内心深处的某个地方是否会想过，这个声明对一个内核模块而言，它到底意味着什么，如果没有它，加载这样一个模块对系统又会造成什么样的影响，如此等等，读者都可以在阅读本书的过程中找到答案。

很显然，只有当你清楚地理解了一个东西的内在机制，你才能更好地去使用它们，如果不幸在使用过程中出现问题，也才可以快速将其定位并最终予以解决。台湾著名技术作家侯捷曾引林雨堂先生在《朱门》中的一句话，“只用一样东西，不明白它的道理，实在不高明”，来描述他当时写作时的心境，其实这句话也同样适合我用来阐明写作本书的初衷之一。

但是这并不意味着只有 Linux 系统下设备驱动程序的编写老手才适合阅读本书，因为我在本书写作过程中，一般会先给出一个总体的框架，然后在此基础上对 Linux 提供给设备驱动程序使用的每一个常见而重要的内核设施进行细致地分析，同时辅之以验证性质的代码来使得这种略嫌抽象的讨论具体化，以激发读者对技术探索的兴趣。所以即便是入门级的读者，也可以通过阅读本书来加深对 Linux 下编写设备驱动程序的理解。

另外要说的是，读者不应该寄希望于阅读两三本书就可以掌握 Linux 下设备驱动程序编写的精髓，所有的书籍只能在大体上给你一个参考借鉴的作用，真正的理解还要靠读者自己去努力，诚所谓“纸上得来终觉浅，绝知此事要躬行”。

编排与范围

在本书的结构编排上，我努力使各章节独立起来，但是少量的向前或者向后引用还是必不可少的。但是总体上，我将最基本的篇章尽量放到前面，一些加强型或者高级点的话题尽量放到后边。在描述驱动程序内核机制方面，为了避免单纯的代码解释所带来的抽象感，我会使用具体的例子来将所能看到的驱动程序的前台行为和它的幕后机制串联起来，以帮助读者建立起全面立体的设备驱动程序架构蓝图。不过 Linux 对某些特性的支持因为考虑到各种平台和性能等诸多因素，其实现很可能会有多种不同的方法，比如从内核态驱动程序向用户空间导出信息的文件系统方面，就至少有 proc 和 sysfs 两种形式。因此本书在描述具体的例子时，一定是遵循其中的某种实现，在诸多实现机制的选择上，本书会从实用性和实时性角度出发，采用内核中最新引入或者是最有发展前景的实现，对于某些即将过时的实现机制（因为兼容或者代码维护工作量的关系，一些老的机制可能依然残留在新版的内核代码中），除非出于技术细节的对比或者从增加知识面的角度考虑会有所涉及，否则将不会作为本书的主线。

在代码的引用上，为了突出功能主线部分和削减本书的篇幅，我会删除代码中用来增加调试信息、性能增强及防御性代码这些部分。对于系统体系架构相关的代码，我主要以 x86 与 ARM 平台为主，因为这两者是当前最流行的两种处理器架构。关于本书所参考的 Linux 内核源码的版本，在本书刚开始写作时参考的是 2.6.35 的版本，在写作的中后期，已经将内核版本更新到了 2.6.39，在本书的修订阶段，我已经努力将之前完成的内容更新到了 2.6.39。当然，因为作者时间精力所限，加之 Linux 内核本身就博大精深，内核版本也一直在不断更新变化中，所以书中肯定还会有这样那样潜在的错误，希望读者朋友们能不吝批评指正，以使我们得以共同提高。

创作历程

我有幸自参加工作以来，在 Linux 下从事设备驱动程序相关的开发工作已经有 9 年多的时间，这期间在 Linux 上所接触的平台既有 x86，也有 ARM，甚至包括少量的 PowerPC。在我看来，学习某一操作系统下的设备驱动程序的编写，主要包含两个方面：一个是该操作系统本身对设备驱动程序框架的支持，也可以称之为设备驱动模型，另一个则是对要驱动的硬件的理解。对于后者，设备驱动程序开发者将要面对各种各样的硬件设备，了解它们的最好也最直接的方法当然是这样硬件的 datasheet。前者则主要和操作系统息息相关，比如在 Linux 系统下开发设备驱动程序，必然要熟练掌握 Linux 为设备驱动的编写所提供的

各种内核实施及相关的各种数据结构，本书的内容主要就是探讨 Linux 内核为设备驱动程序编写所提供的所有这些设施的幕后技术。

本书最早的写作酝酿大约在 2010 年 10 月份前后，在此之前，或者是出于自己对以往积累的技术总结的需要，或者是出于将自己的一些技术心得与同行分享的目的，总之，我陆陆续续在一些论坛上发表了若干剖析 Linux 设备驱动程序内核机制的帖子，这些帖子最终使我萌发了用一本书来总结自己以往的 Linux 设备驱动程序开发经验的想法。我把最初的大约一章半的稿子发给了电子工业出版社，很快就得到了策划编辑张春雨先生的肯定，接下来也很顺利地通过了选题的论证，这之后就是一段极其漫长且非常辛苦的写作过程。时间是最大的挑战，由于白天需要工作，写作的时间只能是留给夜晚或者周末，在写作最紧张的时刻，经常要写到凌晨 2 点多。除了时间上的困难之外，如何将一个技术点用最透彻最简洁的语言描述清楚，如何对 Linux 内核中纷繁复杂的内容进行取舍，这些也都是非常耗费精力的事情。技术本身的理解也许并不困难，难在如何去把你心中掌握的东西清晰准确地以文字的方式表达出来，这不同于论坛的发帖，可以非常自由甚至随心所欲，写书的话，必须考虑它的完整性、逻辑性以及可读性，同时还要考虑将来潜在的读者群。尤其是如果你想认认真真写一本书的话，有时候甚至需要反复推敲一个技术点的表达方式。在写作灵感枯竭的时候，看着时间飞快掠过，而眼前的文档却没有留下几行字，那种强烈的挫折感与沮丧感真得会让人动摇自己的信念：自己是否还能坚持下去？！所以当这本书即将出版时，我还很有些恍惚，不敢相信自己居然磕磕绊绊地最终完成了这些书稿。

意见反馈

读者如果在阅读本书的过程中有任何意见或者建议，欢迎通过下面的 E-mail 与我取得联系：ricard_chen@yahoo.com。

关于本书使用到的源代码，读者可在 www.embexperts.com 网站上下载。另外，关于本书后续的一些勘误、某些技术细节方面的讨论也会在该网站相应的版面上进行。

致谢

首先，我要感谢我的家人，如前所述，写书占去了我大量的业余时间，我的父母和怀孕的妻子在此期间承担了几乎所有的家务劳动，替我捣腾出不少的写作时间，感谢她们！我的宝贝女儿在今年 8 月 15 日健康出生，成为我的家庭中新的一员，这本书也正好可以作为父亲的见面礼送给她——可爱的萌萌同学。

其次是电子工业出版社的张春雨与白涛编辑，从选题的论证到文字编辑，他们都付出了极其辛苦的劳动并且提出了很多有益的建议，那些逝去的不堪回首岁月里满眼尘封的 E-mail 见证了这一点！当然，还要感谢我现在所效力的 AMD 公司，因为它使得我不必为生活所

迫去写一本书，对技术的热情与兴趣才是我最终得以坚持下来的最大因素。

最后，在本书的审核方面，AMD 的 PMTS 及显卡驱动软件架构师俞辉在百忙中为本书作序并审核了部分章节，AMD 上海研发中心 Linux Graphic Base Driver 团队的 Lisa Wu 及研发经理刘刚也为本书的写作提供了支持，诺基亚与西门子的研发经理胡兵全审核了本书第 1 章及第 12 章，EMC 的 PE Thomas 审核了本书第 3 章及第 4 章。Marvell 的资深软件工程师 James Lai 亦审核了本书部分章节并有宝贵意见，在此一并表示感谢！

陈学松

2011 年 8 月 29 日于上海

目　录

第 1 章

内核模块

模块最大的好处是可以动态扩展应用程序的功能而无须重新编译链接生成一个新的应用程序映像，这种广义上的模块概念其实并非 Linux 系统所特有，在微软的 Windows 系统上动态链接库 DLL（Dynamic Link Library）便是模块概念的一个典型应用场景，对应到 Linux 系统上这种模块以所谓的共享库 so（shared object）文件的形式存在[1]。

本章要讨论的主题——Linux 内核模块，在概念及原理方面与上面提到的 DLL 和 so 模块类似，但又有其独特的一面，内核模块可以在系统运行期间动态扩展系统功能而无须重新启动系统[2]，更无须为这些新增的功能重新编译一个新的系统内核映像。内核模块的这个特性为内核开发者开发验证新的功能提供了极大的便利，因为像 Linux 这么庞大的系统，编译一个新内核并重新启动将浪费开发者大量的时间。

虽然设备驱动程序并不一定要以内核模块的形式存在，并且内核模块也不一定就代表着一个设备驱动程序，但是内核模块的这种特性似乎注定是为设备驱动程序而生。Linux 系统下的设备驱动程序员在开发一个新的设备驱动的过程中，使用的最多的工具之一是 insmod，这就是一个简单的向系统动态加载内核模块的命令。很难想象，如果没有 insmod 这样的机制，在 Linux 底下调试一个设备驱动会是怎样的一件让人痛苦抓狂的事情！笔者相信，任何一个在 Linux 上面有过实际的驱动程序开发经历的人都会有类似的感受。

Linux 系统虽然为内核模块机制提供了完善的支持，使得其下的内核模块是如此强大，然而现实中事情往往并非如预想的那样一帆风顺，如果对其幕后的机制不甚了解，在实际的开发过程之中，除了驱动程序自身要实现的功能可能会遇到麻烦以外，在利用 Linux 中的内核模块机制时，也会遇到各种各样的问题，比如在用 insmod 命令加载一个模块时，就很可能会碰到类似下面的错误信息：

```
root@AMDLinuxFGL:/# insmod demodev.ko
```

---

1 在软件工程中，模块这一术语在不同的上下文环境中有不同的语义。本书中提到的模块特指某种动态或者静态链接库。因为静态库在原理上和动态库有很大的区别，所以本章提到的模块，背后都暗含着动态链接的思想，更具体地，就是以.ko 形式存在的所谓内核模块。

2 如果因为动态加载的模块自身的原因导致系统崩溃，则是另一回事了。

insmod: error inserting 'demodev.ko': -1 Invalid module format

如果 dmesg 一下，就会看到内核针对上述错误打印出的出错信息如下所示：

demodev: version magic '2.6.39 SMP mod_unload 586' should be '2.6.39 SMP mod_unload modversions 586'

直觉上，这应该不是在驱动程序自身要实现的功能上出现了问题，问题应该出在驱动程序所在的模块在加载时与系统中内核模块框架互动的环节中。很明显，Linux 内核设计中为模块这种机制提供了完善的支持，以内核模块形式存在的设备驱动程序也必然要遵循这种框架下的规则才能正常工作，也许绝大多数情况下模块都会工作得很好，然而诸如上面提到的这类模块相关的错误也绝非罕见。

既然规则不由我们定义，那么了解并遵守规则就成了避免或者解决这类问题的唯一途径。一个成熟的 Linux 设备驱动程序开发者应该能很快确定这些错误的原因并给出相应的解决方案，而新手在这类错误面前更多的感觉则可能是迷惘和不知所措。因此，无论是出于现实工作的需要，还是为了满足自己的好奇心，Linux 下的设备驱动程序员都有必要花上足够多的时间来了解隐藏在内核模块背后的技术细节，而这也正是本章要深入探讨内核模块机制的目的。本章将重点关注并讨论如下的问题：

- 模块的加载过程。
- 模块如何引用内核或者其他模块中的函数与变量。
- 模块本身导出的函数与变量如何被别的内核模块所使用。
- 模块的参数传递机制。
- 模块之间的依赖关系。
- 模块中的版本控制机制。

1.1 内核模块的文件格式

以内核模块形式存在的驱动程序，比如 demodev.ko，其在文件的数据组织形式上是 ELF（Executable and Linkable Format）格式，更具体地，内核模块是一种普通的可重定位目标文件。用 file 命令查看 demodev.ko 文件，可以得到类似如下的输出：

dennis@AMDLinuxFGL:/$ file demodev.ko
demodev.ko: ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV), not stripped

ELF 是 Linux 下非常重要的一种文件格式，常见的可执行程序都是以 ELF 的形式存在。本