信息科学与技术丛书

冯国进　编著

Linux 驱动程序开发实例

第2版

- 基于 Linux 4.5 内核与 ARM 嵌入式系统
- 全面剖析 Linux 驱动程序开发精髓
- 涵盖多种硬件接口驱动程序
- 附赠完整实例源代码

机械工业出版社
CHINA MACHINE PRESS

信息科学与技术丛书

Linux 驱动程序开发实例

第 2 版

冯国进　编著

机械工业出版社

设备驱动程序是应用程序与硬件设备之间的桥梁，驱动程序开发是软硬件结合的技术。本书深入介绍 Linux 设备驱动程序开发，涵盖了 Linux 驱动程序基础、驱动模型、内存管理、内核同步机制、I2C 驱动程序、串口驱动程序、LCD 驱动程序、网络驱动程序、USB 驱动程序、输入子系统驱动程序、块设备驱动程序、音频设备驱动程序等内容。全书以实例为主线，是为 Linux 设备驱动程序开发人员量身打造的学习书籍和实战指南。本书基于 Linux 4.5 内核，提供了丰富的实例代码和详细的注释，并附赠完整源代码供读者下载。本书主要面向各种层次的嵌入式 Linux 软硬件开发工程师，也可以作为各类嵌入式系统培训机构的培训教材和高校计算机课程教辅书籍。

图书在版编目（CIP）数据

Linux 驱动程序开发实例 / 冯国进编著. —2 版. —北京：机械工业出版社，2017.5
（信息科学与技术丛书）
ISBN 978-7-111-56706-6

Ⅰ. ①L… Ⅱ. ①冯… 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（2017）第 092047 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
责任编辑：车 忱
责任校对：张艳霞
责任印制：李 昂

三河市国英印务有限公司印刷

2017 年 7 月第 2 版 • 第 1 次印刷
184mm×260mm • 27 印张 • 652 千字
0001—3000 册
标准书号：ISBN 978-7-111-56706-6
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

电话服务
服务咨询热线：（010）88361066
读者购书热线：（010）68326294
（010）88379203

网络服务
机 工 官 网：www.cmpbook.com
机 工 官 博：weibo.com/cmp1952
教育服务网：www.cmpedu.com
金 书 网：www.golden-book.com

出 版 说 明

随着信息科学与技术的迅速发展，人类每时每刻都会面对层出不穷的新技术和新概念。毫无疑问，在节奏越来越快的工作和生活中，人们需要通过阅读和学习大量信息丰富、具备实践指导意义的图书来获取新知识和新技能，从而不断提高自身素质，紧跟信息化时代发展的步伐。

众所周知，在计算机硬件方面，高性价比的解决方案和新型技术的应用一直备受青睐；在软件技术方面，随着计算机软件的规模和复杂性与日俱增，软件技术不断地受到挑战，人们一直在为寻求更先进的软件技术而奋斗不止。目前，计算机和互联网在社会生活中日益普及，掌握计算机网络技术和理论已成为大众的文化需求。由于信息科学与技术在电工、电子、通信、工业控制、智能建筑、工业产品设计与制造等专业领域中已经得到充分、广泛的应用，所以这些专业领域中的研究人员和工程技术人员越来越迫切需要汲取自身领域信息化所带来的新理念和新方法。

针对人们了解和掌握新知识、新技能的热切期待，以及由此促成的人们对语言简洁、内容充实、融合实践经验的图书迫切需要的现状，机械工业出版社适时推出了“信息科学与技术丛书”。这套丛书涉及计算机软件、硬件、网络和工程应用等内容，注重理论与实践的结合，内容实用、层次分明、语言流畅，是信息科学与技术领域专业人员不可或缺的参考书。

目前，信息科学与技术的发展可谓一日千里，机械工业出版社欢迎从事信息技术方面工作的科研人员、工程技术人员积极参与我们的工作，为推进我国的信息化建设做出贡献。

机械工业出版社

前　言

写作背景

自 1991 年问世以来，Linux 操作系统一直在创造着开源世界的神话，它已经在服务器、嵌入式系统、智能手机等领域大放异彩，当之无愧地成为了当前最重量级的操作系统。从最初的 Linux 0.01 版到现在的 Linux 4.x 版，让我们看到了 Linux 强大的生命力。我们有理由相信，Linux 操作系统将健康地发展下去。

自十多年前在 Linux 平台上开发第一个应用开始，我便喜爱上了 Linux 平台上的软件开发。从那之后，我有幸能够长期从事嵌入式 Linux 的驱动与应用开发，今后也将在 Linux 驱动开发领域持续耕耘。Linux 带给我无穷的乐趣，我也希望向读者介绍 Linux 平台的驱动开发技术，为 Linux 的发展贡献一点绵薄之力。本书上一版出版之后，很多热心读者发来建议，也促使我创作本书第 2 版。

设备驱动程序依然是 Linux 这个伟大的操作系统的最重要的部分，设备驱动程序开发也是实际项目中非常重要的任务。设备驱动程序关系到系统的稳定可靠，这就要求工程师具备严谨的工作态度。设备驱动程序开发是软件与硬件相结合的领域，希望读者能先了解一些硬件方面的知识，为学习本书打下基础。

“操千曲而后晓声，观千剑而后识器。”我始终认为要成为一个领域的专家，就需要长时间不断地练习以及总结，在实践中不断深入探索是最便捷的学习方法，所以本书遵循了实例驱动的学习模式。希望读者能够认真钻研每一个例程，并举一反三，早日成为一名合格的驱动开发工程师。

本书特点

- 实战性：本书提供多达三十多个驱动程序例程，非常适合各种层次的驱动程序开发人员。书中例子全部基于 Linux 4.5.2 内核。本书附赠代码包含了书中大部分实例的相关代码，读者可以免费下载。
- 全面性：本书涵盖了 Linux 驱动程序基础、驱动模型、内存管理、内核同步机制、I2C 驱动程序、LCD 驱动程序、网络驱动程序、USB 驱动程序、输入子系统驱动程序、块设备驱动程序、音频设备驱动等内容，是驱动程序开发人员的完整参考书。
- 易读性：本书以实例为主线，代码注释丰富，带领读者由浅入深掌握 Linux 驱动程序开发的精髓。

内容结构

本书内容丰富全面，涵盖了 Linux 4.5 下的三类驱动设备，包括字符设备、块设备、网络设备的开发技术。本书第 1～5 章为 Linux 驱动程序开发入门基础知识；第 6 章介绍基本的硬件设备驱动开发；第 7～15 章介绍各种硬件接口的驱动程序体系，包括 I2C、LCD、USB、输入设备、网络、TTY、音频等接口。

读者对象

本书是一本专门介绍嵌入式 Linux 驱动程序开发的书，读者应具备 C 语言编程和操作系

统方面的基础知识。本书主要面向嵌入式 Linux 系统的内核、设备驱动程序、应用程序的开发工程师以及 ARM 嵌入式系统的硬件设计工程师，也可以作为各类嵌入式系统培训机构的培训实验教材和高校操作系统课程的辅导书籍。

特别致谢

在朋友、家人和机械工业出版社的帮助和支持下，本书终于得以问世，在此对他们表示衷心的感谢。特别是责任编辑车忱老师，在本书编写过程中提出了大量合理的建议，使本书得以顺利出版。

本书大部分例程基于深圳友坚恒天的 idea6410 开发板，在此对他们表示特别的感谢。

本人希望能够和读者一起努力，扩大交流，共同进步。由于 Linux 驱动程序开发相当博大精深，加之本人水平有限，本书错误在所难免，请各位读者原谅并指正。读者可把修改建议发送到 fgjnew@163.com，以便再版时修正。

冯国进

2016 年 10 月 1 日

目　录

第 1 章　Linux 设备驱动程序入门

到目前为止，设备驱动程序代码仍然是 Linux 内核代码中最多的部分。操作系统最重要的功能之一就是支持各类设备的访问，承担硬件与应用软件之间的桥梁作用。Linux 操作系统中主要包含字符设备、块设备、网络设备等三类基本的设备驱动程序，内核中的设备驱动程序大部分基于这三类设备驱动。本章主要介绍 Linux 设备驱动程序的入门知识，包含模块基础、字符设备驱动、proc 文件系统等内容。

1.1　设备驱动程序基础

1.1.1　驱动程序的概念

所谓设备驱动程序，就是驱使设备按照用户的预期进行工作的软件，它是应用程序与设备沟通的桥梁。从本质上讲，设备驱动程序主要负责硬件设备的参数配置、数据读写与中断处理。Linux 的运行空间分为内核空间与用户空间。为了保护系统的安全，这两个空间各自运行在不同的级别，不能相互直接访问和共享数据。Linux 内核为应用层提供了一系列系统调用接口，应用程序可以通过这组接口来获得操作系统内核提供的服务。应用层程序运行在用户态，而设备驱动程序是操作系统的一部分，运行在内核态。应用程序要控制硬件设备，首先通过系统调用访问内核，内核层根据系统调用号来调用驱动程序对应的接口函数来访问设备。图 1-1 说明了 Linux 驱动程序的运行原理。

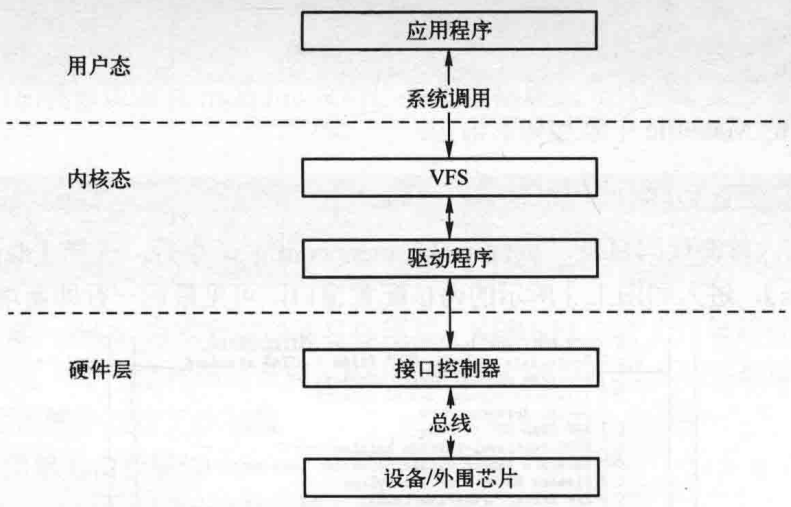

图 1-1　设备驱动程序的原理

Linux 中的大部分驱动程序是以内核模块的形式编写的。内核模块是 Linux 内核向外部提供的一个接口，其全称为动态可加载内核模块（Loadable Kernel Module，LKM）。Linux

内核本身是一个单内核（monolithic kernel），具有效率高的优点，也具有可扩展性和可维护性差的缺陷。模块机制就是为了弥补这一缺陷而生。内核模块可以被单独编译，它在运行时被链接到内核作为内核的一部分在内核空间运行。要让内核支持可加载模块，需要配置内核的【Enable loadable module support】选项，如图 1-2 所示。

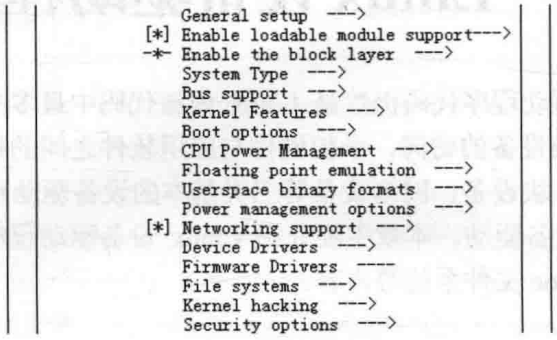

图 1-2　在内核中增加可加载模块支持

1.1.2　驱动程序的加载方式

Linux 设备驱动程序有两种加载方式。一种是直接编译进 Linux 内核，在 Linux 启动时加载；另一种是采用内核模块方式，这种模块可动态加载与卸载。

如果希望将新驱动程序编译进内核，需要修改内核代码和编译选项。下面以字符型设备为例，说明如何在 Linux 内核中添加一个新的设备驱动程序。如果驱动程序代码源文件为 infrared_s3c2410.c，将 infrared_s3c2410.c 复制到内核代码的/drivers/char 目录，并在该目录下的 Kconfig 文件最后增加如下语句：

```
config INFRARED_REMOTE
    tristate "INFRARED Driver for REMOTE"
    depends on ARCH_S3C64XX || ARCH_S3C2410
    default y
    help
```

在该目录下的 Makefile 中添加如下语句：

```
Obj-$(CONFIG_INFRARED_REMOTE)+= infrared_s3c2410.o
```

进入 Linux 内核源代码目录，执行 make menuconfig 命令后，选择【device drivers】->【character devices】，进入如图 1-3 所示的内核配置窗口，可见最后一行即新增的驱动：

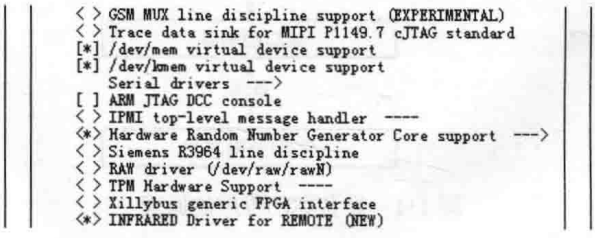

图 1-3　在内核中增加新驱动程序

在内核配置窗口中可以使用上下键、空格键和回车键进行选择、移动和取消选择。内核配置窗口中以< >带头的行是内核模块的配置，以[ ]带头的行是内核功能的配置。选项前如果为<*>，表示相应的模块将被编译进内核。如果选项前是< >则表示不编译进内核。这里在【INFRARED Driver for REMOTE】行前面设置为<*>，则 infrared_s3c2410.o 将被编译进内核。在使用 make zImage 命令编译内核时所有设置为<*>的项将被包含在内核映像中。

采用可加载模块方式让驱动程序的运行更加灵活，也更利于调试。可加载模块用于扩展 Linux 操作系统的功能。使用内核模块的优点是可以按照需要进行加载，而且不需要重新编译内核。这种方式控制了内核的大小，而模块一旦被插入内核，它就和内核其他部分一样，可以访问内核的地址空间、函数和数据。可加载模块通常以.ko 为扩展名。在图 1-3 中选项前如果为<M>，表示编译成可加载模块。在使用 make modules 命令编译内核时，所有设置为<M>的项将被编译。make modules 结束后可以使用下面的命令安装内核中的可加载模块文件到一个指定的目录：

```
make modules_install INSTALL_MOD_PATH=/home/usr/modules
```

使用 make 命令编译内核相当于执行 make zImage 和 make modules 两个命令。

1.1.3　编写可加载模块

Linux 内核模块必须包含以下两个接口：

```
module_init(your_init_func);        //模块初始化接口
module_exit(your_exit_func);        //模块卸载接口
```

加载一个内核模块的命令是 insmod，格式如下：

```
#insmod modulename.ko
```

卸载一个内核模块的命令是 rmmod，格式如下：

```
#rmmod modulename
```

可加载模块的源代码可以放在内核代码树中，也可以独立于内核代码树。如果是后一种情况，需要为可加载模块编写 makefile 文件。可加载模块的 makefile 文件最重要的就是设置如下几个变量：

```
CC= arm-none-linux-gnueabi-gcc
obj-m:= smodule.o
KERNELDIR ?= /root/fgj/linux-4.5.2
```

CC 是编译器，obj-m 为需要编译的目标模块，KERNELDIR 为内核路径。注意在编写可加载模块前先要有一个内核代码目录树。KERNELDIR 的内核版本必须与运行的内核版本一致，否则编译出的模块往往无法加载。

例 1.1　最简单的内核模块

代码见\samples\1door\1-1simple。核心代码如下：

```
static int demo_module_init(void)
{
    printk("demo_module_init\n");
```

```
    return 0;
}
static void demo_module_exit(void)
{
    printk("demo_module_exit\n");
}
//模块入口
module_init(demo_module_init);
module_exit(demo_module_exit);
MODULE_DESCRIPTION("simple module");
MODULE_LICENSE("GPL");
```

模块运行在内核态，不能使用用户态 C 库函数中的 printf 函数，而要使用 printk 函数打印调试信息。编写一个 Makefile 文件如下：

```
AR = ar
ARCH      = arm
CC = arm-none-linux-gnueabi-gcc
DEBFLAGS = -O2
obj-m       := smodule.o
KERNELDIR ?= /root/fgj/linux-4.5.2
PWD         := $(shell pwd)
modules:
    $(MAKE) -C $(KERNELDIR) M=$(PWD) LDDINC=$(PWD)/../include modules
clean:
    rm -rf *.o *~  core .depend .*.cmd *.ko *.mod.c .tmp_versions
```

执行 make 后生成 smodule.ko。运行结果如下：

```
[root@urbetter drivers]# insmod   smodule.ko
demo_module_init
[root@urbetter drivers]# cat /proc/modules
smodule 868 0 - Live 0xbf000000 (O)
[root@urbetter drivers]# rmmod smodule
rmmod: can't change directory to '/lib/modules': No such file or directory
[root@urbetter /home]#mkdir -p /lib/modules/'uname -r'
[root@urbetter drivers]# rmmod smodule
demo_module_exit
```

第一次运行 rmmod smodule 会失败，因为需要在/lib/modules 目录下建立以内核版本号为名称的目录，才能正确卸载模块。uname –r 用来得到内核版本号。建立正确的目录后，模块可以正常卸载。

1.1.4 带参数的可加载模块

宏 MODULE_PARM(var,type,right) 用于向模块传递命令行参数。参数类型可以是整数、长整型、字符串等类型。