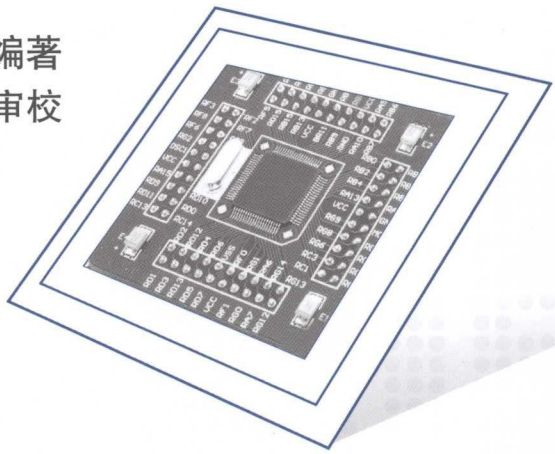
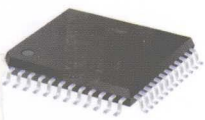
普通高校“十二五”规划教材 · 实践创新系列

嵌入式LINUX驱动程序实战开发

奚海蛟　谌　利　吕铁军　编著
达内IT培训集团　审校





北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

普通高校“十二五”规划教材·实践创新系列

嵌入式 Linux 驱动程序实战开发

奚海蛟　谌　利　吕铁军　编著
达内 IT 培训集团　审校

北京航空航天大学出版社

内 容 简 介

驱动程序是连接上层应用层和底层硬件层之间的桥梁，负责直接对硬件进行操作，是嵌入式系统开发中不可或缺的重要组成部分。本书以 SAMSUNG 公司的 S3C2440 为代表的 ARM9 为核心，以广州天嵌科技有限公司开发的 TQ2440 为硬件平台，讲述了驱动程序的开发流程和必备知识，并针对 TQ2440 开发板的各个外设接口，为读者提供了简单而直观的驱动程序实例，以加深读者的理解。本书可分为两部分：一是驱动程序开发的基础，二是驱动程序开发实例。通过理论和实践相结合，使读者更容易掌握。

本书可供嵌入式 Linux 驱动程序开发的人员、使用 S3C2440 进行快速开发产品的开发人员参考，还可作为各大中专院校和培训机构的教材。

图书在版编目(CIP)数据

嵌入式 Linux 驱动程序实战开发/ 奚海蛟，谌利，
吕铁军编著. -- 北京 ：北京航空航天大学出版社，2012.10
ISBN 978-7-5124-0925-5

Ⅰ. ①嵌… Ⅱ. ①奚… ②谌… ③吕… Ⅲ. ①
Linux 操作系统—程序设计 Ⅳ. ①TP316.89

中国版本图书馆 CIP 数据核字(2012)第 207666 号

嵌入式 Linux 驱动程序实战开发

奚海蛟 谌 利 吕铁军 编著

达内 IT 培训集团 审校

责任编辑 李松山

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191) http://www.buaapress.com.cn

发行部电话：(010)82317024 传真：(010)82328026

读者信箱：emsbook@gmail.com 邮购电话：(010)82316936

涿州市新华印刷有限公司印装 各地书店经销

*

开本：710×1 000 1/16 印张：22.25 字数：487 千字

2012 年 10 月第 1 版 2012 年 10 月第 1 次印刷 印数：4 000 册

ISBN 978-7-5124-0925-5 定价：45.00 元

本书编委会

主　编：奚海蛟
副主编：韩少云

编委会成员

刘张辉　冯　华　谌　利　张　泉　李政春
李宝栋　滕忠楠　孟　捷　刁为民　游成伟

前　言

随着嵌入式行业的迅猛发展，嵌入式开发也已经成为当前最热门、最有发展前途的行业之一，同时，嵌入式行业的快速发展造成了它巨大的人才缺口。越来越多的人抓住这个机遇投身到嵌入式行业中。

随着嵌入式行业的发展，“嵌入式操作系统”这个名词也开始被各行各业人所熟知，嵌入式的应用也在逐渐改变着我们的生活。相信，在不久的将来，随着嵌入式产品的广泛应用，我们的生活必将进入一种全新的状态。

说到“嵌入式操作系统”就不能不说 Linux 操作系统。Linux 操作系统以其高效性和灵活性著称，在嵌入式系统中占有举足轻重的地位。而且其开放的源代码更适合于嵌入式操作系统需要裁剪的特性，因此，Linux 系统被广泛地应用于嵌入式设备中。而在嵌入式系统中，Linux 设备驱动程序占据着重要的地位。对于嵌入式设备来说，其硬件具有可裁剪性，不同的嵌入式设备所支持的硬件可能不同，但是它们却使用的都是 Linux 系统，那么相应硬件的驱动开发在其中起到了关键性作用。

Linux 设备驱动程序是连接系统内核和硬件设备的桥梁，是该硬件在 Linux 操作系统下可以正常运行的必要条件。因此，在嵌入式设备的研发过程中，驱动程序的研发是承上启下的部分。本书以 Linux 驱动程序开发为核心，以 TQ2440 开发板为硬件平台，向读者讲述了具体硬件平台上的驱动程序的开发过程，其中包括环境的搭建、驱动的开发和驱动的测试。本书以实战开发为主线，读者可以更加深刻地体会驱动程序的开发流程，可以更好地理解驱动开发所必备的知识。

本书组织结构

第 1 章是 Linux 设备驱动的概述，介绍设备驱动的作用、Linux 设备驱动的分类、用户空间和内核空间的划分以及驱动程序的编译方法。通过本章的学习，读者已经开始对 Linux 设备驱动程序有了一个初步的概念。

第 2 章详细介绍开发环境的构建，是读者能够更好地完成接下来的实战开发所必不可缺少的部分。只有完成了本章的环境搭建，所编写的驱动程序以及测试程序才能顺利地移植到目标平台 TQ2440 开发板。

第 3 章是本书的一个核心，是能够顺利完成下面几个章节驱动程序实战开发的知识储备。本章是 Linux 设备驱动开发基础，介绍字符设备驱动程序的框架、竞争并发的处理机制、阻塞 I/O 和异步通知机制以及时间度量方法。最终通过一个简单的 LED 驱动开发实例结尾，可以让读者对驱动程序的结构框架有一个初步的认识。通过本章的学习，读者可以对驱动程序框架有一个初步认识，可以掌握驱动程序中所要使用的必要的处理机制。

第 4 章介绍 Linux 设备驱动的调试方法，使用正确的调试方法，可以让用户的驱动程序开发更加直观，可以准确地定位问题的所在，节省开发时间，是驱动开发过程中必

不可少的部分。

第5章主要讲述驱动程序设计中的中断处理机制。本章通过TQ2440开发板的按键驱动程序实例，讲述驱动程序设计中中断的申请、处理和释放的过程。以简单的驱动程序实例说明了复杂的中断处理机制。通过本章的学习，读者可以更加深刻地理解Linux系统中的中断处理。

第6章主要讲述A/D转换器的工作原理以及SPI总线的通信原理。本章通过A/D转换器和SPI通信的综合实例为读者展示了A/D转换的原理以及SPI总线上主从设备的通信原理。

第7章主要讲述Linux中的终端设备的驱动程序。本章通过S3C2440的串口驱动程序实例为基础向读者展示了Linux终端设备的工作原理、S3C2440串口通信所使用的寄存器以及相关寄存器的配置选项。

第8章主要讲述I2C总线的通信原理。本章通过实例向读者展示了S3C2440中I2C总线的特点以及通信原理。

第9章主要介绍RTC时钟驱动程序的开发。本章通过RTC设备驱动程序实例，介绍了Linux下的时钟系统、时间表示方法和S3C2440中的RTC控制器。

第10章主要介绍触摸屏驱动程序的开发。目前，触摸屏作为输入子系统的一部分已经广泛的应用于嵌入式设备之中。本章通过S3C2440触摸屏实例讲解了触摸屏的工作原理、S3C2440的触摸屏接口以及Linux的input子系统驱动架构。

第11章主要讲述Linux设备驱动分类中网络设备驱动的开发。本章通过DM9000网卡驱动的实例向读者讲解了网络设备的工作原理以及设计流程。

第12章主要讲述现在使用最广泛的一类通用串口USB的设备驱动程序。本章分别分析了主机USB和设备USB的工作原理，以及如何在Linux系统中添加USB驱动程序。

读者对象

想要学习或者从事嵌入式Linux驱动程序开发的人员。

使用S3C2440进行快速开发产品的开发人员。

尽管本书面向触及嵌入式系统开发人员，但是阅读本书需要熟悉相关的硬件知识以及C语言和汇编等编程语言。

致 谢

参与本书编写的主要人员有刘张辉、冯华、李政春、张泉、吴飞、陈露露、李宝栋、杨帆、滕忠楠、李晓庆、付盈、孟捷、谌利和游成伟等。由奚海蛟博士后和达内IT培训集团总裁韩少云负责全书的规划、内容安排、定稿和修改。北京达内IT集团总裁韩少云、北京航空航天大学出版社相关人员、广州天嵌计算机科技有限公司梁传智对本书的出版给与了极大的支持，在此向他们表示衷心的感谢。

作 者

2012年8月

目　　录

第 1 章　Linux 设备驱动概述

Linux 设备驱动在整个 Linux 嵌入式系统中占据很重要的地位，它是连接内核和硬件设备的桥梁。了解 Linux 设备驱动基本知识，掌握开发的基本方法，是整个驱动学习的基础。本章将对 Linux 设备驱动开发的方法进行介绍，通过本章的学习，读者可以了解到 Linux 设备驱动程序的基本概念以及如何编写驱动程序。

本章要点：

- 设备驱动的作用；
- Linux 设备驱动的分类 ；
- 内核空间和用户空间；
- 编译驱动程序。

1.1　设备驱动的作用

不仅仅是嵌入式系统，就是我们正在使用的 PC，也是硬件和软件的合集。众所周知，对于硬件来说，如果不对其进行软件的设计，也就是说如果不编写程序去驱使它工作，那硬件只是一堆废铁而已。对于驱动而言，其本质就是让硬件能够正常工作的代码。硬件为底层实物基础，软件为上层应用逻辑，硬件和软件相辅相成才能成为所用的仪器、系统等。

当然，对于驱动程序来说，其不仅仅是一部分软件这么简单，更形象地说驱动程序在系统中拥有的地位，就好比是一座桥梁，它连通了硬件和软件。而且，它也区分了 3 部分人的工作，那就是：硬件工程师的任务就是设计性能优越的硬件产品，像 CPU、内存、显示卡、显示屏、各种接口等；而软件工程师，他们在各种操作系统中，编写各种实用有趣的应用程序，为我们的工作和生活提供方便；这两者各司其职，但却是完全独立的两个领域，软件工程师不必了解硬件工程师的工作，而硬件工程师也不参与软件工程师的软件设计，就好像刚刚说到的，软件工程师编写的软件是运行在硬件上的，那怎么保证软件能够正确地让硬件工作，实现软件的功能呢？这就到驱动工程师大显身手的时候了，很好理解，驱动工程师编写硬件设备的驱动程序，这些程序封装在系统中，为软件工程师对硬件的操作提供接口，如在我们熟知的 Windows 系统中，操作系统就提供了许多 API 函数，让应用程序的编写者能够调用这些 API 函数，如要让显示器显示，就调用显示的 API，让声卡发出声音，就调用发声的 API，硬件设备驱动程序在这其中就占据了这样的地位，起到了这样的作用。

在没有操作系统的设备中，工程师可以根据硬件的特性自己定义接口，然而在有操作系统的情况下，设备驱动的架构就由操作系统决定，在设计设备驱动的时候必须依照相应操作系统的架构来进行，这样设备驱动才能良好地整合到操作系统的内核中。

作为沟通硬件和软件的桥梁，设备驱动的重要性不言而喻。随着通信、电子行业越来越迅速的发展，芯片、电路模块的日新月异，硬件的飞速更新赋予了设备驱动的开发工作更为重要的意义。这些被开发的设备驱动，或运行在简单的单任务环境中，或运行在复杂的(如 Linux、Windows 等)多任务操作系统环境中，发挥着不可替代的重要作用。

1.2 Linux设备驱动的分类

通常，计算机系统的硬件主要分为 CPU、存储器和外设，驱动针对的对象是存储器和外设，包括 CPU 内部集成的和 CPU 外部的，Linux 将存储器和外设分为以下 3 类。

- 字符设备(character device)：如键盘、鼠标、串口等。
- 块设备(block device)：如硬盘、Flash 等。
- 网络接口(network interface)：如以太网等。

1. 字符设备

字符设备是指在 I/O 传输过程中以字符为单位进行传输的设备，如键盘、打印机等。就是说它的读/写都是以字节为单位的。例如，串口在收发数据时，是一个字节一个字节进行的，键盘在输入按键时也是一个键值一个键值传递的。在 Linux 系统中，字符设备的驱动程序实现了 open、close、read、write 等系统调用，可以使用与普通文件相同的文件操作命令对字符设备文件进行操作。字符设备是最基本、最常用的设备。概括地说，字符设备的驱动主要完成以下 3 件事：

- 定义一个结构体 static struct file_operations 变量，其内定义一些设备的 open、close、read、write 等控制函数；
- 在结构体外分别实现结构体中定义的这些函数；
- 向内核中注册或删除驱动模块。

2. 块设备

块设备是与字符设备并列的概念，这两类设备在 Linux 系统中驱动的结构有较大差异。总体而言，块设备驱动比字符设备驱动要复杂得多，在 I/O 操作上表现出极大的不同，缓冲、I/O 调度、请求队列等都是与块设备驱动相关的概念。在块设备上，数据以块的形式存放，如 NAND Flash 上的数据就是以页为单位存放的。块设备驱动程序向用户层提供的接口与字符设备一样，应用程序也可以通过相应的设备文件(如/dev/mtdblock0、/dev/hda1 等)来调用 open、close、read、write 等系统，与块设备传送任意字节的数据。也就是说，对用户而言，字符设备和块设备访问方式没有差别。块设备驱动程序的特别之处在于：

① 操作硬件的接口实现方式不一样。

这是什么意思呢？块设备驱动程序先将用户发来的数据组织成块，再写入设备；或者从设备中读出若干块数据，再从中挑选出用户所需要的。这和字符设备以一个字节一个字节传送的方式不同。

② 数据块上的数据可以有一定的格式。

通常，在块设备中按照一定的格式存放数据，所谓的不同的文件系统类型就是用来定义这些格式的。内核中，文件系统位于块设备驱动程序层之上，这意味着块设备驱动程序除了向用户提供与字符设备一样的接口之外，还要向内核的其他部件提供一些接口，这些接口对于用户而言是不可见的，它们使得可以在块设备上挂载文件系统。

3. 网络接口

网络接口是区别于字符设备和块设备的第三大标准类设备，它和前两种设备不同，因为 UNIX 世界里“一切皆是文件”的论述对于它来说并不适用。例如，块设备可以在系统文件树的/dev 目录下找到特定的文件入口标志，而网络设备则没有这种文件操作入口。UNIX 式的操作系统访问网络接口的方法是给它们分配一个唯一的名字（如 eth0），而这个名字在文件系统中（如刚刚提到的/dev 目录下）不存在对应的节点项。网络接口同时具有字符设备、块设备的部分特点，但是都有不同。对比于字符设备，它的输入/输出是有结构的、成块的（报文、包、帧）；对比于块设备，它的“块”又不是固定大小的，可以大到数百甚至数千字节，又可以小到几个字节。由于网络接口并不是以文件的方式存在，这就导致应用程序、内核和网络驱动程序间的通信完全不同于字符设备和块设备，内核提供了一套 push 等操作来完成数据包的转换与递送，而不是 open、read、write 等。

1.3　内核空间与用户空间

对于驱动开发来说，不可不知内核空间和用户空间的概念。理解了内核空间和用户空间的概念和作用，对理解系统的体系结构有很大帮助。

在 386 以后的计算机体系结构中，都区分了内核空间和用户空间的保护机制。提供保护的目的，是要避免系统中的任务访问另外的或属于操作系统的存储区域。在 IntelX86 体系中，提供了特权级这种保护机制，通过特权级别的不同来限制对存储区域的访问。Linux 操作系统也是基于这种架构，对自身进行了划分：一部分核心软件独立于普通应用程序，包括驱动程序，运行在较高的特权级别上，它们驻留在被保护的内存空间，拥有访问硬件设备的所有权限，这部分就被称为内核空间；相对地，其他部分被作为应用程序在用户空间被执行，用户空间的应用程序只能看到允许它们使用的部分系统资源，并且不能使用某些特定的系统功能，不能直接访问硬件，如果特殊情况下需要对内核空间进行访问，也必须得到内核空间的授权方可。

这些所谓的空间其实就是内存的地址，内核空间的页表是常驻内存的，不会被虚拟

内存管理模块换出到磁盘上，内核空间的程序一旦出错，系统就会死机。而应用程序在某些情况下，得到内核的审核授权，也可以提高其权限，对内核空间的部分资源进行访问。这样区分不仅使管理更合理，也在很大程度上提高了系统的可靠性，为系统提供了保护机制。

内核空间和用户空间都是指虚拟空间，也就是虚拟地址，目前 32 位系统支持 4 GB 的虚拟地址空间。在 Linux 系统中，4 GB 虚拟地址空间的最高 1 GB 地址被分配给内核使用，即内核空间，被内核独享，而低地址的 3 GB 空间为用户共享。

1.4　编译驱动程序

2.6 版本内核驱动大多都编译为.ko 模块。在 2.6 内核中要想编译模块，必须先存在已经成功编译了的内核源码树，而且该源码树编译出来的内核就是该模块即将运行在其上的内核。

1）编译方法 1

（1）将写好的 hello.c 驱动程序文件放入/opt/EmbedSky/linux-2.6.30.4/drivers/char/目录下，

然后修改同目录下的 Kconfig 文件，在其中加入 Hello 驱动的配置单如下：

```
config Hello
    tristate "Hello Driver"
    depends on ARCH_S3C2440
    help
    S3C2440 Hello.
```

然后，修改同目录的 Makefile 文件，在其中加入下面这行：

```
obj-$(CONFIG_Hello) += Hello.o
```

添加完以上内容之后，输入：

```
#make menuconfig
```

然后配置内核驱动的配置单如下，在配置单中选中刚刚加入的驱动程序：

首先选择 Device Drivers，如图 1.1 所示。

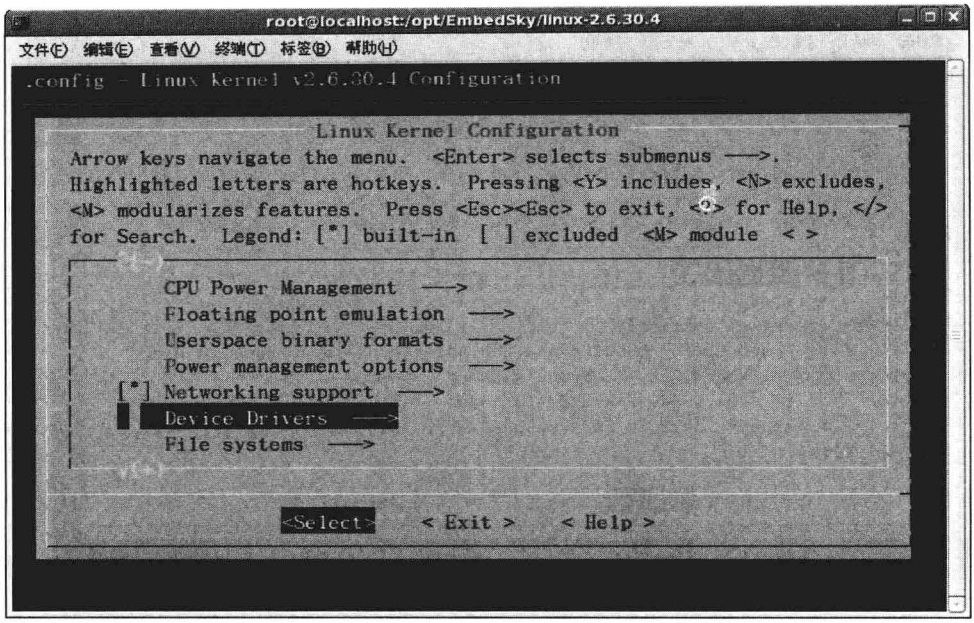


图 1.1　选择 Device Drivers

然后选择 Character devices，如图 1.2 所示。

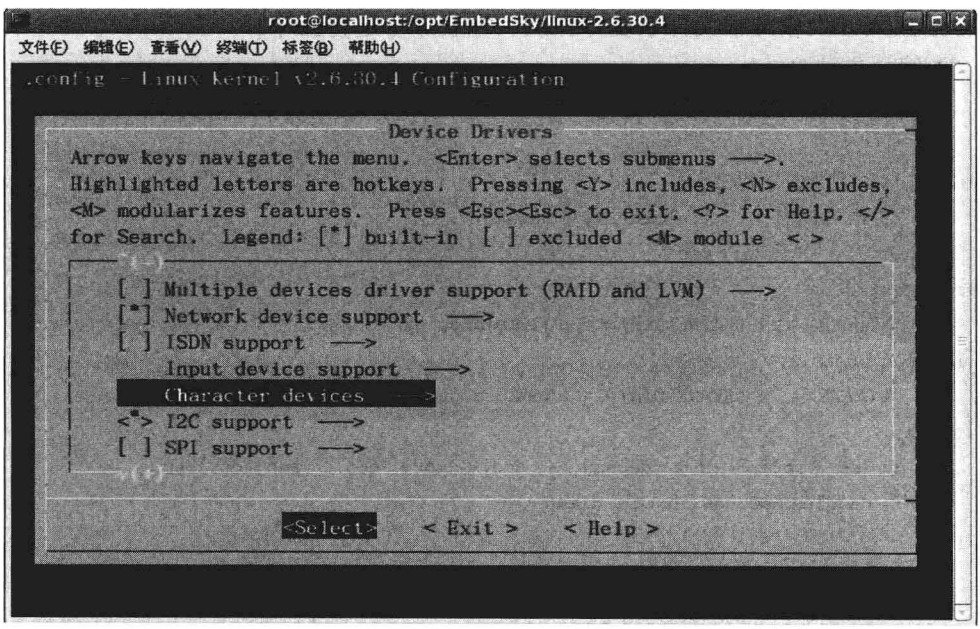


图 1.2　选择 Character devices

最后选择＜＊＞Hello Driver，如图 1.3 所示。

将其选择为“＊”(添加到内核中)，然后保存配置。

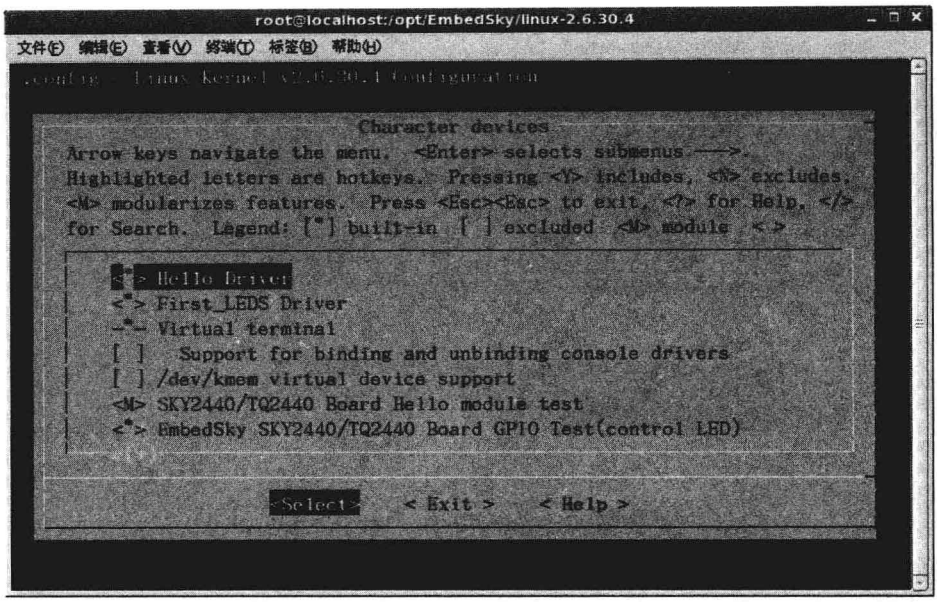


图 1.3　选择 Hello Driver

值得注意的是，如果此处将其选择为“M”(模块)，然后保存配置，接下来还需要重复上面所说的方法，执行命令“#make SUBDIR=drivers/char/ modules”，生成 hello.ko，然后放入开发板的根文件系统，用加载、卸载命令进行操作。

2) 编译方法 2

(1) 在.c 文件当前目录下编写 Makefile：

```
ifeq ($(KERNELRELEASE),)
    KERNELDIR ? = /opt/EmbedSky/linux-2.6.30.4
    PWD := $(shell pwd)
modules:
    $(MAKE) -C $(KERNELDIR) M=$(PWD) modules
modules_install:
    $(MAKE) -C $(KERNELDIR) M=$(PWD) modules_install
clean:
    rm -rf *.o *~ core .depend .*.cmd *.ko *.mod.c .tmp_versions
.PHONY: modules modules_install clean
else
    obj-m := hello.o
endif
```

(2) 在.c 文件当前目录下执行编译命令 make，然后生成模块 hello.ko。

注意：当 make 时，由于变量 KERNELRELEASE 没有被赋值，因此 ifeq ($(KERNELRELEASE),)为真，于是变量 KERNELDIR 被赋值为内核源码树目录/