嵌入式 Linux 下 PCI 设备驱动的设计与实现*

袁爱平
（长沙民政职业技术学院 软件学院，湖南 长沙 410004）

摘　要：PCI 局部总线具有使用方便、数据传输率高等特点，已成为计算机必备的一种接口。Linux 是一种日趋成熟完善的操作系统，越来越多的软硬件厂商开始使用 Linux 平台开发自己的产品，因而对基于该平台的设备驱动程序的需求也愈来愈多。介绍了 Linux 驱动程序开发的一般方法，并实现了流媒体数据缓存 PCI 卡在 Linux 环境下的驱动程序。

关键词：Linux 操作系统；PIC 总线；设备驱动；流媒体数据缓存卡
中图分类号：TP316　　**文献标识码**：B　　**文章编号**：1674-7720(2011)12-0009-02

Design and implement of PCI device driver based on Linux

Yuan Aiping
(College of Software, Changsha Social Work College, Changsha 410004, China)

Abstract: Nowadays PCI bus has become a universal bus with its characteristics of easy using and high data transfer rate. With the development of Linux, there are more and more companies to develop their productions under Linux operation system .So there are more and more requirements to develop the devices driver for hardware under Linux. This paper introduces a method to develop device drivers based on Linux, and implement the streaming media data cache PCI card driver for Linux.

Key words: Linux OS; PCI bus; device driver; streaming media data cache PCI card

随着通用处理器和嵌入式技术的迅猛发展，越来越多的电子设备需要由处理器控制。目前大多数 CPU 和外部设备都会提供 PCI 总线的接口，PCI 总线已成为计算机系统中一种应用广泛、通用的总线标准[1]。Linux 因其开放源代码以及稳定的性能，越来越受到广大用户青睐。同时，基于 Linux 内核的嵌入式操作系统应用势头强劲，开发基于 Linux 的设备驱动程序，具有很强的实用性和可移植性[2]。

1 PCI 总线概述

PCI(Peripheral Component Interconnect)总线，即外部设备互连，是现在流行的一种连接 PC 和外围设备的总线结构[3]。PCI 提供了一组完整的总线接口规范，可以在 33 MHz 时钟频率、32 bit 数据总线宽度的条件下达到峰值 132 Mb/s 的传输速率；它能支持一种称为线性突发的数据传输模式，可确保总线不断满载数据；采用总线主控与同步操作，显著改善 PCI 的性能；PCI 独立于处理器的结构，用户可随意增添外围设备，以扩展电脑系统而不必担心在不同时钟频率下会导致性能下降。

2 PCI 设备驱动程序的设计与实现

Linux 中将设备分成字符设备、块设备和网络设备三种类型，通过主设备号和从设备号实现对设备的描述。其中主设备号描述控制该设备的驱动程序，即驱动程序与主设备号一一对应，从设备号用来区分同一个驱动程序控制的不同设备[5]。

PCI 设备属于字符设备。本设计采用模块方式实现 PCI 卡驱动程序。驱动程序主要由设备注册和注销、设备探测和移除、设备中断处理和系统调用等函数组成。

2.1 设备注册和注销

使用一个设备之前，必须保证已经对它进行注册，这项工作一般是在设备初始化时完成。设备初始化函数中调用函数 register_chrdev( )来注册字符设备。流媒体数据缓存 PCI 卡驱动程序的注册代码如下：

```
#define MAJOR_NUM 128
register_chrdev(MAJOR_NUM,"pci_card",&pci_card_fops);
```

将设备的主设备号设为 128，设备名称为 pci_card。pci_card_fops 是一个 file_operations 结构指针，这个结构是设备驱动程序所提供的入口点位置，在设备注册时向系统进

* 基金项目：湖南省科技计划项目(2008GK3085)

行登记，以便系统在适当时调用。pci_card_fops 定义如下：

```
static struct file_operations pci_card_fop={
    owner:THIS_MODULE,
    open:pic_card_open,
    release:pic_card_release,
    read:pic_card_read,
    write:pic_card_write,
    ioctl:pic_card_ioctl
};
```

当不再使用此设备时，需调用 unregister_chrdev( )函数注销驱动程序。

2.2 设备探测和移除

在扫描到新的 PCI 设备后，系统需要调用设备驱动程序实现的探测函数以查找与设备相匹配的 PCI 驱动。流媒体数据缓存 PCI 卡设备驱动的探测函数 pic_card_probe()的主要实现代码如下：

```
pci_card = kmalloc(sizeof(struct pci_card), GFP_KERNEL);
                                    //为设备实例分配存储空间
pci_enable_device(dev);             //激活 PCI 设备
spin_lock_init(pci_card ->lock);
                                    //初始化特定设备实例的私有化数据
pci_read_config_byte(dev, PCI_REVISION_ID, (u8*)&(pci_
    card ->rev_id));                //读取配置信息
pci_card->mem_base=pci_resource_start(dev, 0);
                                    //读取 I/O 资源的配置信息
pci_request_regions(dev, "pic_card");   //申请 I/O 区域
pci_set_master(dev);                //设置成总线主模式
pic_card->mem_start=ioremap(pic_card->mem_base,
    pic_card->mem_size);            // I/O 内存映射
```

设备移除函数主要完成释放映射的虚拟地址、释放 I/O 区域、关闭 PCI 设备和释放为设备实例分配的内核空间等功能。

2.3 中断处理

流媒体数据缓存卡驱动中的中断处理程序主要负责识别中断、响应中断和唤醒睡眠的进程，中断处理代码如下：

```
inl(pci_card->iobase+PCI_CARD_INT_STA); // 识别中断
outl(status&INT_MASK, pci_card->iobase + PCI_CARD_
INT_STA);                           //响应中断
wake_up_interruptible(&pci_card->wq);   //唤醒睡眠进程
```

2.4 系统调用

用户进程利用系统调用对设备文件进行诸如 read/write 操作时，系统调用通过设备文件的主设备号找到相应的设备驱动程序，然后读取这个数据结构相应的函数指针，接着把控制权交给该函数。流媒体数据缓存 PCI 卡的系统调用函数主要包括设备的打开、关闭、读写和控制等。

在使用 PCI 设备之前，必须先打开所要使用的 PCI 设备。当用户在应用程序中调用 open()函数时，应用程序就会自动进入驱动程序中的 pci_card_open () 函数。pic_card_open()函数主要负责增加模块的使用计数，并根据 pic_card_probe()读到的中断号申请中断，注册中断处理程序。具体实现如下：

```
MOD_INC_USE_COUNT
request_irq(pci_card->irq, pci_card_interrupt, SA_SHIRQ, "
pci_card", pci_card));
```

在使用完 PCI 设备后，必须关闭 PCI 设备。当用户在应用程序中调用 close()函数时，应用程序就会自动进入驱动程序中的 pci_card_release()函数。pci_card_release()函数的主要工作是释放中断和减少模块的使用计数。

用户在应用程序中调用 read()函数和 write()函数对设备文件进行读写操作时，应用程序就会自动进入驱动程序中的 pci_card_read()函数和 pci_card_write()函数。pci_card_read()函数首先会阻塞在以 pci_card->wq 为队头的等待队列上。当流媒体数据缓存卡上的数据准备好，即 pci_card->state 变为 READY 时，pci_card_read()函数会被唤醒。函数被唤醒后，会先将数据从设备 I/O 内存拷贝到内核空间，再从内核空间拷贝给用户进程，实现方式如下：

```
wait_event_interruptible (pci_card->wq, pci_card->state==
READY);
memcpy_fromio(pbuf, pci_card->mem_start, count);
copy_to_user(buf, pbuf, count));
```

而 pci_card_write()函数的主要工作是将数据从用户进程拷贝到内核空间，再将内核空间中的数据拷贝到设备 I/O 内存，实现代码如下：

```
copy_from_user(pbuf, buf, count);
memcpy_toio(pci_card->mem_start, pbuf, count);
```

Linux 是一种日趋成熟完善的操作系统，PCI 总线已成为计算机系统中一种应用广泛、通用的总线标准。本文针对流媒体数据缓存卡设备，结合 PCI 总线的特点，开发实现了流媒体数据缓存 PCI 卡在 Linux 环境下的设备驱动程序，本文介绍的驱动原理同样适用其他 PCI 设备的开发。

参考文献

[1] 陈颖，唐超. 基于 PCI 总线驱动程序设计方法研究[J]. 微计算机信息，2008，12(1)：272-274.

[2] 李善平，刘文峰，王焕龙. Linux 与嵌入式系统[M].北京：清华大学出版社，2003.

[3] 宋有泉，高小鹏，龙翔. 嵌入式 PCI 网卡驱动程序的设计与优化[J]. 计算机工程，2007，3(2)：264-266.

[4] 王峰，张文军，余松煜. PCI 设备驱动程序中几个关键问题的设计与实现[J]. 测控技术，2002，21(8)：58-60.

[5] 钱晨，徐荣华，王钦若. 基于 Linux 操作系统的设备驱动程序开发[J]. 微计算机信息，2004，20(9)：131-133.

(收稿日期:2011-02-28)

作者简介：

袁爱平，男，1975 年生，硕士研究生，软件设计师，主要研究方向：嵌入式系统、图形图像处理。