嵌入式 Linux 系统中触摸屏控制的研究与实现

李中奇 张冬波 华东交通大学电气与电子学院(330013)
罗文俊 华东交通大学基础科学学院 (330013)

Abstract

This paper mainly discusses the research and application of embedded system base on Linux.Firstly a summarization about embedded system is given,then a plan that develop device driver for touch screen based on an embedded Linux operating system.The design of its device driver and applications for Linux embedded operating system is introduced in detail.

Keywords:embedded system,Linux,touch screen,driver program

摘 要

本文主要讨论了基于嵌入式 Linux 操作系统的研究与开发。文章首先对嵌入式系统进行了简单介绍,在详细分析了系统特点的基础上,结合 Linux 自身的优点,提出了基于嵌入式 Linux 操作系统对触摸屏驱动的开发方案。并详细介绍了驱动程序及测试应用程序的设计。

关键词:嵌入式系统,Linux,触摸屏,驱动程序

目前,流行的商用嵌入式操作系统主要有 Windows CE、Vxworks、PSOS、QNX 等。这些专用操作系统均属于商业化产品,价格昂贵,且源代码不公开,使得每个系统上的应用软件和其它系统都不一样。嵌入式 Linux 的出现打破了这一僵局。它是可以进行裁剪、修改使之能在嵌入式计算机系统上运行的一种操作系统。既继承了 Internet 上无限的开放源代码的资源,又具有嵌入式操作系统的特性。它具有稳定性和安全性、良好的硬件支持、标准兼容性和资源丰富等功能。触摸屏是一种方便、快捷的输入设备,附着在显示器的表面,与显示器配合使用。在工业控制场合的到了广泛的应用。本文根据实际的基于嵌入式 Linux 的电力机车受电弓检测仪的人机接口,论述了在这一系统中如何对触摸屏的控制。

1 Linux 下的设备驱动

Linux 将设备分为最基本的两大类,字符设备和块设备。字符设备是以单个字节为单位进行顺序读写操作,通常不使用缓冲技术,如鼠标等。驱动程序实现比较简单,而块设备则是以固定大小的数据块进行存储和读写的,如硬盘,软盘等。为提高效率,系统对于块设备的读写提供了缓存机制,由于涉及缓冲区管理,调度,同步等问题,实现起来比字符设备复杂得多。Linux 的设备管理是和文件系统解密结合的,各种设备都以文件的形式存放在/dev 目录下,称为设备文件。应用程序可以打开,关闭,读写这些设备文件,完成对设备的操作,就像操作普通的数据文件一样。为了管理这些设备,系统为设备编了号,每个设备号又分为主设备号和次设备号。主设备号用来区分不同种类的设备,而次设备号用来区分同一类型的多个设备。对于常用设备,Linux 有约定俗成的编号,如硬盘主设备号是 3。

Linux 为所有文件,包括设备文件提供了统一的操作函数接口。但是对于不同的外设,其操作方式各不相同。在本系统中触摸屏所完成的功能是将感测触点坐标,将坐标值 A/D 转换后传给 CPU。驱动程序需要控制设备的采集并且把数据送往上层的应用程序。以后的处理由应用程序来完成。设备驱动程序要为设备提供通用的系统调用,如 open、read、write、close 等,如图 1。

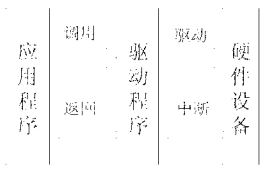

图 1 设备驱动程序和其他部分关系

2 嵌入式 Linux 系统下的驱动程序

Linux 是自由的多任务操作系统,它需要 PC 桌面系统作为运行平台。而本文所讨论的嵌入式 Linux 是指经过小型化裁剪、能够烧录入容量只有几百 KB 或几 MB 的闪存(Flash Memory) 内,不需要硬盘作为存储介质,也不需要键盘、鼠标之类的外设,适用于 8 位/16 位/32 位 MCU,应用于各种特定嵌入式场合的专用 Linux 操作系统。

嵌入式 Linux 设备驱动程序中有一个很重要的数据结构 file_operations{}。它是驱动程序与应用程序的接口,使编写驱动程序的工作变得简单而规律。在该触摸屏驱动程序中定义了一个数据结构为 file_operations{}的变量 touch_fops,并进行了如下的赋值:

```
static struct file_operations touch_fops =
{
        read:           touch_read,
        write:          touch_write,
        open:           touch_open,
        release:        touch_release,
        poll:           touch_poll,
};
```

(1) touch_open 函数

touch_open () 这个函数在 file_operations {} 中的原型是 open()函数。它的主要功能就是打开设备并初始化设备准备进行操作。下面一段程序介绍了 touch_open()函数的实现过程。

```
if((ret=request_irq(IRQ_touchRX,touch_rx,0,"touch_rx",dev_idtouch)
{
    printk("touch_rx_init:failed to register IRQ_touchRX \n");
        free_irq(IRQ_touchRX , dev_idtouch);
        return ret;
}
```

在这个 if 语句中出现了 3 个函数。printk()是 Linux 内核提供的函数,功能近似标准 C 函数库中提的 printf()函数。在 Linux 操作系统中,因为驱动程序是在内核空间运行的,所以必须使用内核提供的函数,print()不能在内核空间运行。

Request_irq()是申请中断的函数,其中参数 IRQ_touchRX 是所申请的中断号,touch_rx 是所安装的中断处理函数,第三个参数是用于中断管理的一些常量,这里的值为 0,表示可以进行

中断的共享，参数“touch_rx”是发送中断的设备名称，dev_id-touch 是用来共享的中断号。如果成功申请中断的话则返回 0 给 ret 变量，当返回了一个非 0 的值给 ret 变量的时候，则说明有另外一个驱动程序已经占用了要申请的中断信号线。当申请中断失败后，就必须进行中断信号线的释放，使用的是 free_irq ()中断信号线释放函数。

(2) touch_read 函数

touch_read()函数的原型是 read()函数。它的作用就是从触摸屏设备中读取数据。当在 file_operations{}结构中用 NULL 来表示此函数的话，则说明这个设备是不允许进行读操作的，如果对其进行调用的话，内核将会返回一个错误。下面来对这个 touch_read()函数进行一下分析。

```
While(rx_user_count>0)
{
        if((USTAT0&HCQ_RX_EMPTY_BIT)==0X40)
        {
            touch_rx_buf[rx_buf_count]=*URXBUF0;
            rx_buf_count++;
        }else {
            interruptible_sleep_on_timeout  (&rx_queue,TIME-
OUT);
}
```

这个循环语句的作用是开辟一个缓冲区用来存放从触摸屏设备中传来的数据。其中 interruptible_sleep_on_timeout()是延时函数，具体是用定时器来进行延时。延时的原因是由于触摸屏设备把数据传入缓冲区需要一定的时间，而 CPU 必须等数据进入了缓冲区后才能进行数据的读取。

(3) touch_write 函数

touch_write()函数的原型是 write()函数，它的主要作用是向触摸屏设备发送数据和命令。原理与 touch_read()函数类似，只是数据传输的方向不同。

(4) touch_release 函数

touch_release()函数的原型是 release()函数。用来关闭触摸屏设备的。如果用 NULL 代替，则表示设备永远是关闭的。这个函数实现起来比较简单，先调用 free_irq()释放触摸屏设备占用的中断控制线，接着使设备的引用计数为 0，这样就完成了对触摸屏设备的关闭工作。

3 触摸屏的应用程序

Linux 操作系统中应用程序工作在用户区。触摸屏应用程序通过已加载到内核模块中的驱动程序控制触摸屏。应用程序可以通过触摸屏实际使用情况来编写。在我们实际的测控系统中触摸屏做为输入设备，与液晶显示屏配合使用达到完成相应的按钮指令的功能。下面是测试触摸屏能否正常工作的应用程序。

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
FILE *fp;
int main()
{
    char read_buf[2];
    char write_buf[2]={GetX, GetY};
    int fd,qr;
    fd=open("/dev/touch",O_RDWR);      //以可读写方式打开前
面加载的触摸屏驱动模块
    if(fd<0)
    {
        printf("Error in oprn Device!\n");  //如打开失败，提示错
误并推出进程
        exit(-1);
    }
    printf("open device ok!\n");
    while(1)
    {
        write(fd,write_buf,1);
        qr=read(fd,read,1);
        if(qr>0)
        {
            GetLocation();
        }
    }
}
void GetLocation()
{
    fp=fopen("/home/touch.txt","w+");
    write(fd,write_buf,1);
    fwrite(read(fd,read_buf,1),1,1,fp);
    write(fd,&write_buf[1],1);
    fwrite(read(fd,read_buf,1),1,1,fp);
    fclose(fp);
}
```

主程序通过 fd=open(“/dev/touch”,O_RDWR)语句进行了一个 open()函数的系统调用，用来调用这个触摸屏驱动程序，并以可读可写的方式来打开触摸屏，把该 open()系统调用的值返回给 fd，作为判断打开该触摸屏是否成功。接下来程序用 while(1)来进行循环检测触摸屏是否有触发动作发生，当有触发动作发生时调用按键处理程序来进行触点位置的读取。

按键处理程序用来得到触点的位置。调用一个写数据的函数向触摸屏控制器发送命令，然后触摸屏根据 CPU 所发送的命令确定返回的坐标数据。这个程序中是先发送读取 X 坐标的命令，然后发送读取 Y 坐标的命令，最后得到触点的坐标值并打印出来。

4 结束语

本文通过介绍在嵌入式 Linux 操作系统下对触摸屏设备驱动的开发过程，使用户了解嵌入式系统设备驱动的开发流程。通过这种方法开发的设备驱动程序可 Linux 操作系统下稳定运行，并取得了良好的效果。

参考文献

1 Touch Screen Controller ADS7846 Datasheet.[J] TI, 2000(9)
2 ALESSANDRO RUBINI. LINUX 设备驱动程序.[M]中国电力出版社，2000
3 陈莉君.Linux 操作系统内核分析.[M]北京：人民邮电出版社，2000
4 黄超.LINUX 高级开发技术.[M]北京：机械工业出版社，2002

[收稿日期：2004.9.6]

我刊《工业控制计算机》关于 2005 年页码增加的通知

随着我刊影响力的日益提高，投稿量也不断增加，由于篇幅的局限，使不少合格的稿件只能退稿处理。应广大读者和作者的要求，2005 年起本刊《工业控制计算机》增加页码，正文由 64 页增加至 80 页。欢迎业界人士积极投稿，投稿注意事项详见我刊网站(www.ipcm.com.cn)“稿件征集”。欢迎到当地邮局或与我社联系订阅，邮局订阅代号：28-60，月刊，订价 6.00 元/期。