设备驱动

121180155 张皓

2014 年 11 月 18 日

目录

1 实验目的

- 学习 Linux 操作系统下内核程序的编写和应用;
- 学习可编程接口芯片的编程控制方法。

2 实验内容与要求

- 完成一个内核模块的编写, 实现内核模块的正确加载和卸载;
- 建立一个虚拟的字符设备驱动程序, 至少要包含读、写功能, 为用户程序提供内核空间与用户空间的数据交换, 方案及实现过程自定;
- 在上面的基础上完成 8253 或 UART 的驱动, 与自编的应用程序结合, 实现特定的功能 (如, 8253 的音乐播放, UART 的双机通信)。

- 思考: 设备文件的文件名、设备名和设备号各起什么作用?
- 查阅参考资料, 画出个人计算机系统中与定时器/计数器有关的电路连接, 或 9 针/25 针串行端口连接器各引脚的信号;
- 考虑采用 8253 进行硬件定时的方案 (暂不用中断实现);
- 思考: 如果用计算机的键盘模拟琴键, 实现类似电子琴的功能, 还需要解决哪些问题?
- 哪些因素影响串行数据传输的可靠性?

3 实验设计思想

3.1 可编程定时器/计数器

早期的个人计算机中, 有一片可编程的定时器/计数器 8253, 作为系统的硬件时钟设备。8253 在系统中占用 40H 43H 端口。三个定时器/计数器的时钟输入均为 1.19MHz. 其中 T/C2, 控制系统的扬声器, 产生声音信号。它的控制端 GATE2 和扬声器前均接有控制信号, 这些控制信号来自可编程 I/O 接口芯片 8255 的 PB0 和 PB1。8255 初始化已将 B 口设为方式 0 输出。

在 8253 提供的六种工作方式中, 只有方式二和方式三是不需要硬件触发能产生连续波形的方式, 其中方式三的输出近似方波。我们可以利用 T/C2 对扬声器的控制功能。如果 T/C2 产生适当频率的方波, 并能将相关的开关打开, 使其作用在扬声器上, 便可以听到该频率的声音

3.2 内核模块

3.2.1 driver.c

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/io.h>
#include <linux/fs.h>
#include <asm/uaccess.h>
#include <linux/ioport.h>
```

```
//定义8253和8255的各端口地址
#define PB 0x61
#define PORT_CTL 0x63
#define TIMER2 0x42
#define TIMER_CTL 0x43

//打开设备
int drv_open(struct inode* inode, struct file* filp)
{
    printk("<1>speaker: dev is open\n");
    //先读写低八位，再读写高八位，设置为方式3，采用二进制计数
    //这个控制字即为10110110B，即0xb6
    outb_p(0xb6, TIMER_CTL);
    //PB1=1，PB0=1，即00000011B，0x03
    outb_p(3, PB);
    return 0;
}


int drv_release(struct inode* inode, struct file* filp)
{
    //PB1=0，PB0=0，关闭扬声器
    outb_p(0, PB);
    printk("<1>speaker: dev is closed\n");
    return 0;
}

//先写低八位，再写高八位
int drv_write(struct inode* inode, char* buf, size_t length, loff_t *offset)
{
    outb_p(buf[0], TIMER2);
    outb_p(buf[1], TIMER2);
    return 0;
}
```

```
//设备文件操作接口定义
struct file_operations fop =
{
    open : drv_open,
    write : drv_write,
    release : drv_release,
};

//初始化模块
int init_module()
{
    if(register_chrdev(240, "speaker", &fop) < 0)
    {
        printk("speaker: register err\n");
        return -1;
    }
    printk("<1>speaker: register success\n");
    return 0;
}

//模块的卸载
void cleanup_module()
{
    unregister_chrdev(240, "speaker");
    printk("<1>speaker: unregister dev\n");
}

MODULE_LICENSE("GPL");
```

3.2.2 Makefile

```
obj-m    =    driver.o
KDIR    =    /lib/modules/`uname -r`/build
```

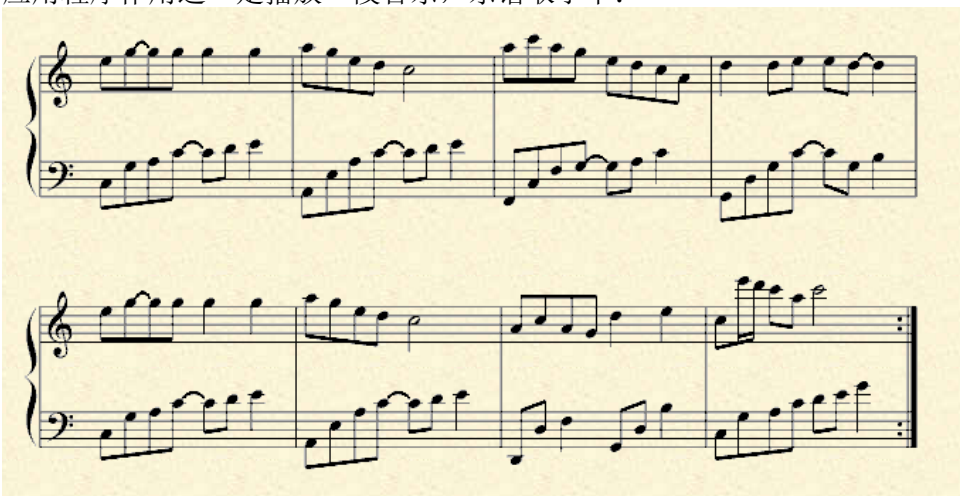
```
PWD       =    `pwd`
default:
    $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
clean:
    $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean

speaker: speaker.c
    gcc -o $@ $^
```

3.2.3 内核模块的加载

```
>make #生成内核模块
#将/sbin添加到PATH路径中
>export PATH=/sbin:$PATH
>insmod driver.ko
#检查添加情况
>lsmod | head
#建立设备节点
>mknod /dev/speaker c 240 0 -m 666
#检查建立情况
>ll /dev/speaker
```

3.3 应用程序

应用程序作用之一是播放一段音乐，乐谱取于下：



3.3.1 speaker.c

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

int main(void)
{
    int fd;
    int i, time[1024];
    //各标准音的频率
   int voice[32] = {0, 131, 147, 165, 175, 196, 220, 247, 0, 0, 0,
                    262, 294, 330, 349, 392, 440, 494, 0, 0, 0,
                   523, 587, 659, 698, 784, 880, 987, 0, 0, 0};
   //一段乐谱, 1-7代表低音1-7,11-17代表中音1-7, 21-27代表高音1-7
   int song[] = {13, 15, 15, 15, 15, 16, 15, 13, 12, 11, 16, 21, 16, 15, 13, 12, 11, 6, 12,
    //各音符的音调
   float tune[] = {0.25, 0.5, 0.25, 0.5, 0.5, 0.25, 0.25, 0.25, 0.25, 0.5, 0.25, 0.25, 0.

    char c;
    long long delay;
    int stop = 0;

    //打开设备文件并错误检查
    if((fd = open("/dev/speaker", O_RDWR)) < 0)
    {
        printf("open dev err\n");
        return -1;
    }

   //********************** test for play a song ********************************
#if 1
```

```
    printf("test for play a song\n");
    for(i = 0; i < 45; ++i)
    {
        //计算计数器初值
        time[i] =  1.192e6 / (1.0*voice[song[i]]);
        write(fd, &time[i], 2);
        //根据音调延时
        usleep(tune[i]*2e6);

        //短间隔
        write(fd, &stop, 2);
        usleep(2e4);
    }
#endif

   //******************* test for keyboard input ********************************
#if 0
    printf("test for keyboard input\n");
    printf("character 1 - 7 is for low 1 - 7\n");
    printf("character q - u is for medium 1 - 7\n");
    printf("character a - j is for high 1 - 7\n");
    //使系统不必每次回车确认输入
    system("stty cbreak");
    //每读入一个字符，判断它属于哪一个音
    while((c = getchar()) != 'c')
    {
        switch(c)
        {
                case '1':
                        i = 1;
                        break;
                case '2':
                        i = 2;
                        break;
```

```
case '3':
        i = 3;
        break;
case '4':
        i = 4;
        break;
case '5':
        i = 5;
        break;
case '6':
        i = 6;
        break;
case '7':
        i = 7;
        break;
case 'q':
        i = 11;
        break;
case 'w':
        i = 12;
        break;
case 'e':
        i = 13;
        break;
case 'r':
        i = 14;
        break;
case 't':
        i = 15;
        break;
case 'y':
        i = 16;
        break;
case 'u':
```

```
                        i = 17;
                        break;
                case 'a':
                        i = 21;
                        break;
                case 's':
                        i = 22;
                        break;
                case 'd':
                        i = 23;
                        break;
                case 'f':
                        i = 24;
                        break;
                case 'g':
                        i = 25;
                        break;
                case 'h':
                        i = 26;
                        break;
                case 'j':
                        i = 27;
                        break;
        }
        //计算计数器初值
        time[i] =  1.192e6 / (1.0*voice[i]);
        write(fd, &time[i], 2);
        sleep(1);
    }
#endif

    close(fd);
    return 0;
}
```

3.3.2 编译及运行

```
>make speaker
>./speaker
```

4 实验分析与小结

本次实验共完成两大功能，分别是播放一段预先写入的音乐和仿真一个电子琴设备。

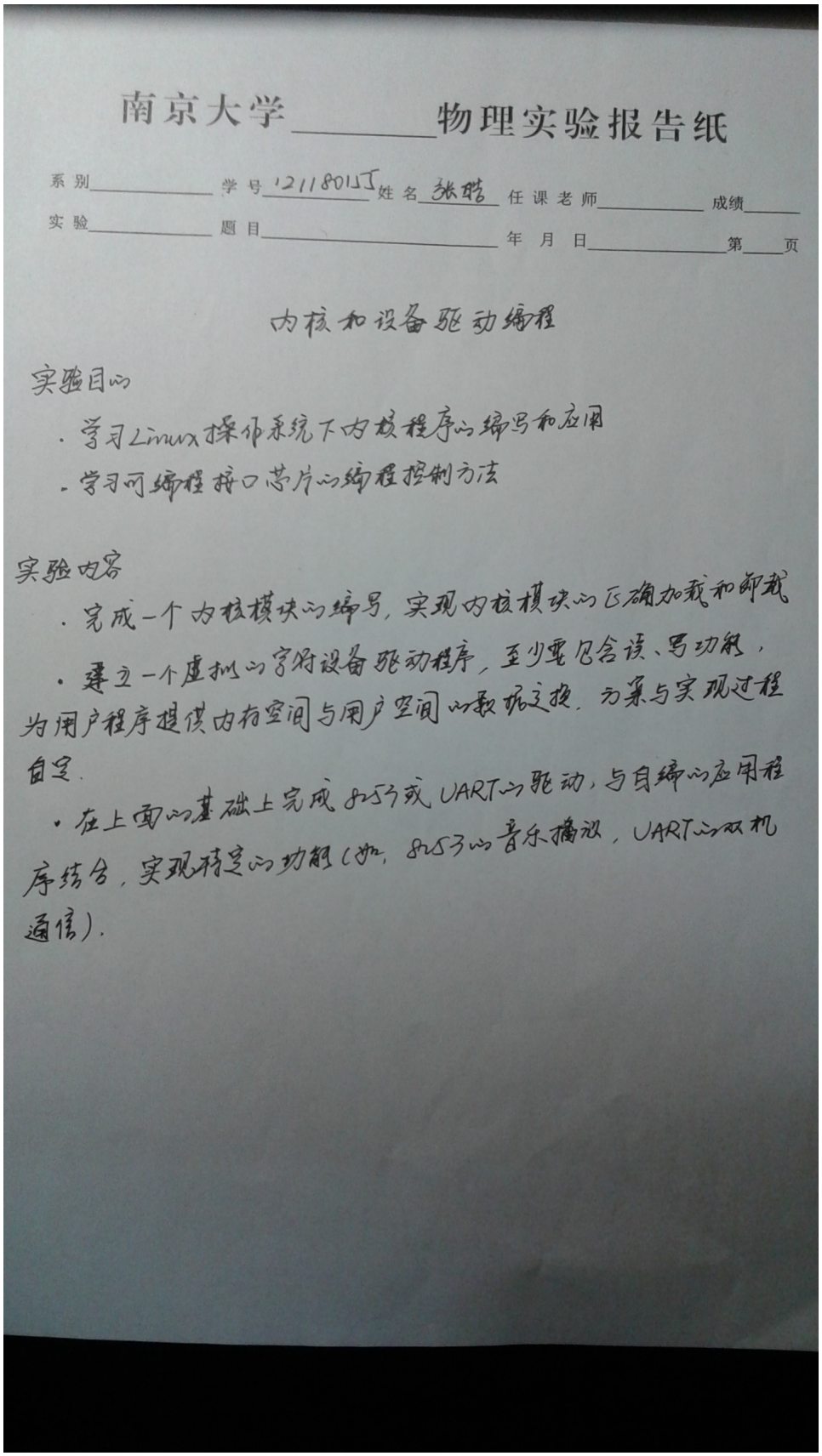
预写入音乐的效果见附件，实现了不同音调的输出。

在模拟电子琴的部分通过 system(”stty cbreak”) 可以免于每次输入字符后还需回车。实验达到了预期的效果。

5 参考文献

- 微处理器与嵌入式系统 (试用教材). 方元, 彭成磊. 南京大学.2014

6 预习报告页面



南京大学_____物理实验报告纸

系别_____ 学号121180113 姓名张皓 任课老师_____ 成绩_____

实验_____ 题目_____ 年 月 日_____第____页

内核和设备驱动编程

实验目的

- 学习Linux操作系统下内核程序的编写和应用
- 学习可编程接口芯片的编程控制方法

实验内容

- 完成一个内核模块的编写，实现内核模块的正确加载和卸载
- 建立一个虚拟的字符设备驱动程序，至少要包含读、写功能，为用户程序提供内存空间与用户空间的数据交换，方案与实现过程自定。
- 在上面的基础上完成8253或UART的驱动，与自编的应用程序结合，实现特定的功能(如，8253的音乐播放，UART的双机通信)。