博芯电子
Prochip
Your Embedded Partner

Linux驱动编写

张黎明

火龙果•整理
uml.org.cn

主要内容：

- 1.Linux驱动简述及字符型驱动的框架
- 2.基于Gpio的Linux字符型驱动设计
- 3.Linux键盘驱动的设计

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

博芯电子
Prochip
Your Embedded Partner

1.Linux驱动简述及字符型驱动的框架

火龙果•整理
uml.org.cn

1.1 什么是设备驱动

- 设备驱动程序是操作系统内核和机器硬件之间的接口.设备驱动程序为应用程序屏蔽了硬件的细节，这样在应用程序看来，硬件设备只是一个设备文件， 应用程序可以象操作普通文件一样对硬件设备进行操作.设备驱动程序是内核的一部分，它完成以下的功能:

  1.对设备初始化和释放.

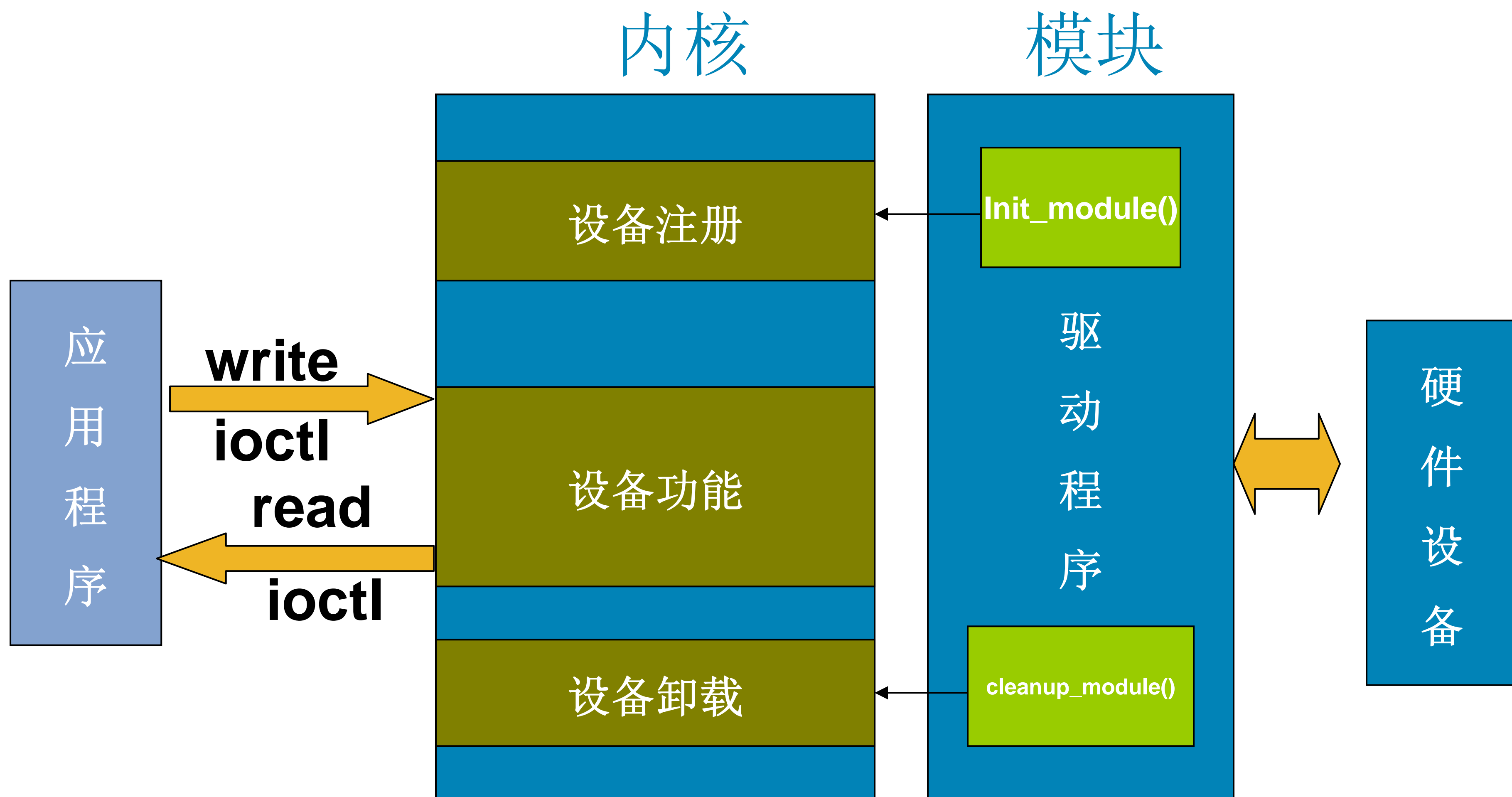
  2.把数据从内核传送到硬件和从硬件读取数据.

  3.读取应用程序传送给设备文件的数据和回送应用程序请求的数据.

  4.检测和处理设备出现的错误.

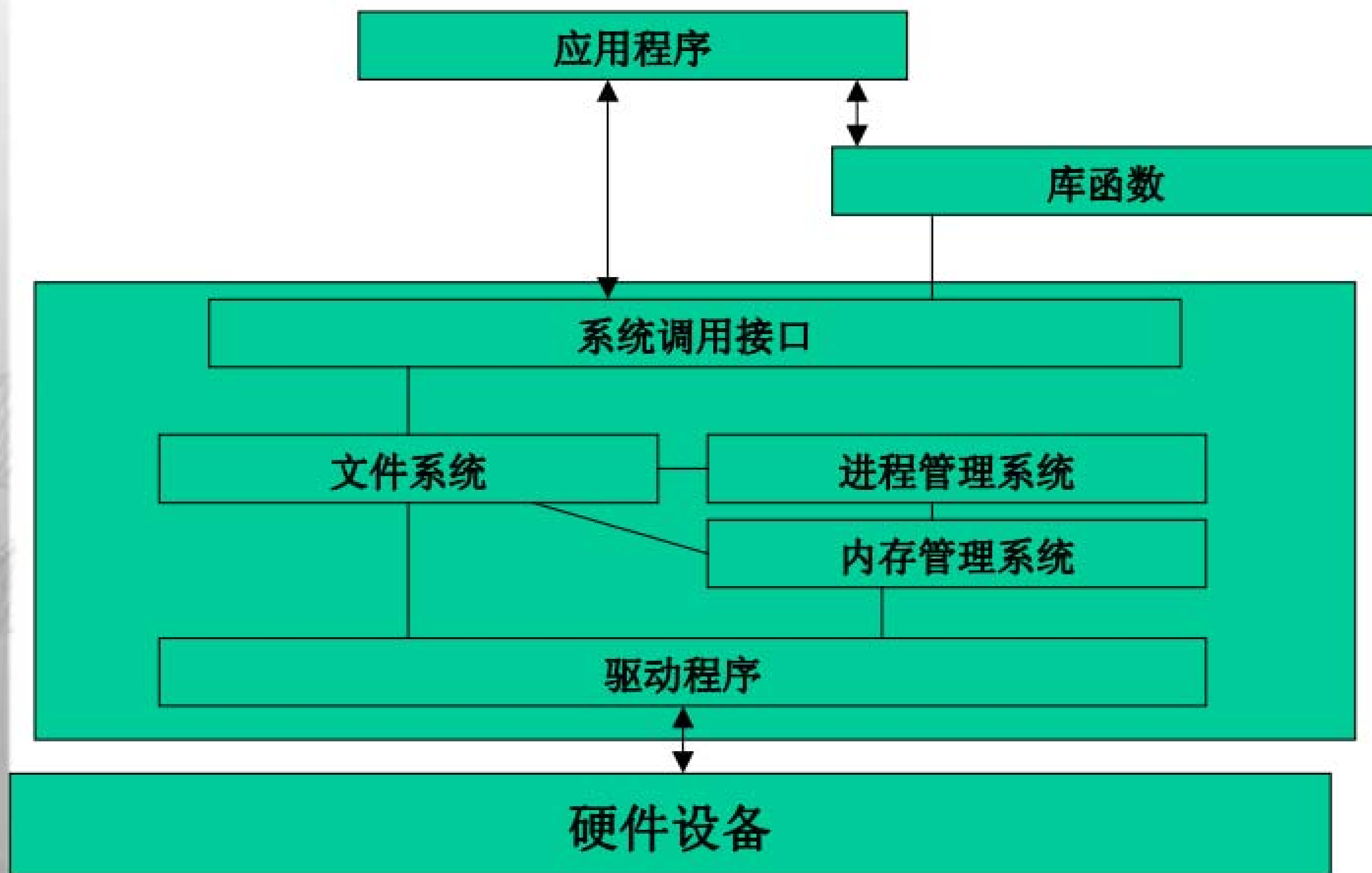
博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

设备，驱动，内核，应用程序之间的调用关系：



博芯电子 Prochip Your Embedded Partner

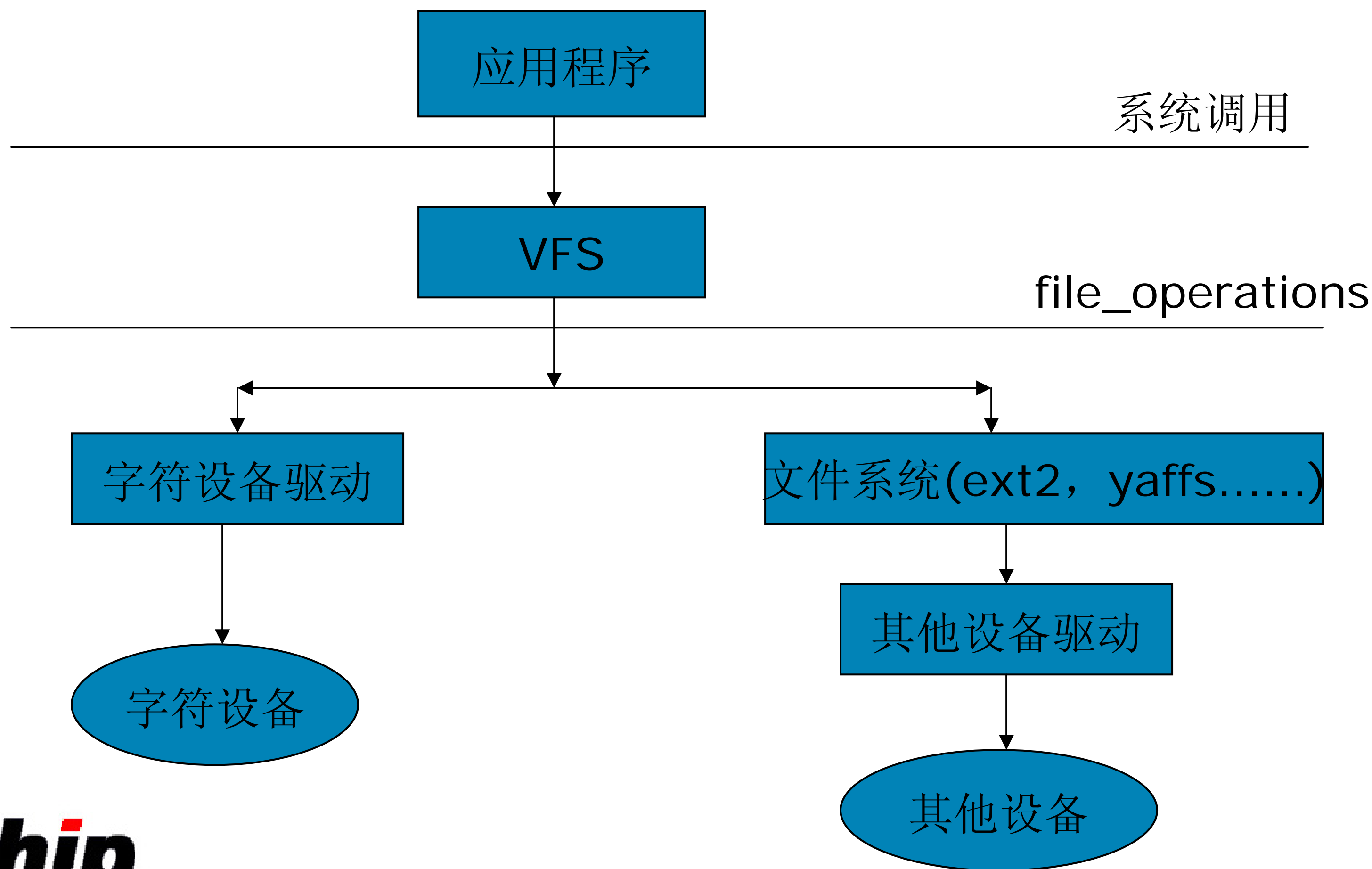
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理 uml.org.cn



Linux内核体系结构

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

设备，驱动，文件系统，应用程序之间的调用关系：



博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

1.2 设备驱动程序特点

- (1)核心代码：设备驱动程序是核心的一部分，像核心中其他的代码一样，出错将导致系统的严重损伤。一个编写不当的设备驱动程序甚至能够使系统崩溃导致文件系统的破坏和数据的丢失；
- (2)标准接口：设备驱动程序必须为Linux核心或者其从属的子系统提供一个标准的接口；
- (3)核心机制：设备驱动程序可以使用标准的核心服务比如内存分配、中断发送和等待队列等；
- (4)动态可加载：多数的Linux设备驱动程序可以在核心模块发出加载请求时进行加载，同时在不使用设备时进行卸载，这样核心可以有效地利用系统的资源
- (5)可配置：Linux设备驱动属于核心的一部分，用户可以根据自己的需要进行配置来选择适合自己的驱动

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

1.3 Linux设备的分类

- 字符设备

  以字节为单位逐个进行**I/O**操作

  字符设备中的缓存是可有可无

  不支持随机访问，如串口设备

- 块设备

  块设备的存取是通过**buffer**、**cache**来进行

  可以进行随机访问

  例如**IDE**硬盘设备

  可以支持可安装文件系统

- 网络设备

  通过**BSD**套接口访问，网络接口 **–** 任何网络事务处理都是通过接口实现的，即，可以和其他宿 主交换数据的设备。通常接口是一个硬件设备，但也可以象 **loopback**（回路）接口一样是软件工具。

  由于不是面向流的设备，所以网络接口不能象**/dev/tty1**那 样简单地映射到文件系统的节点上。**Unix**调用这些接口的方 式是给它们分配一个独立的名字（如**eth0**）。这样的名字在 文件系统中并没有对应项。内核和网络设备驱动程序之间的 通信与字符设备驱动程序和块设备驱动程序与内核间的通信 是完全不一样的。内核不再调用**read**，**write**，它调用与数 据包传送相关的函数。

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理 uml.org.cn

1.4 Linux设备文件的概念

- Linux抽象了对硬件的处理，所有的硬件设备都可以作为普通文件一样来看待
- 可以使用和操作文件相同的、标准的系统调用接口来完成打开、关闭、读写和I/O控制操作，对用户来说，设备文件与普通文件并无区别
- 字符设备和块设备是通过文件节点访问的。在Linux的文件系统中，可以找到（或者使用mknod创建）设备对应的文件名，称这种文件为设备文件。

- 主设备号：标识该设备的种类，也标识了该设备所使用的驱动程序

  Linux内核支持动态分配主设备号
- 次设备号：标识使用同一设备驱动程序的不同硬件设备

  每次内核调用一个设备驱动程序时， 它都告诉驱动程序它正在操作哪个 设备。主设备号和从设备号合在一 起构成一个数据类型并用来标别某 个设备。

  MKDEV(ma,mi)　　　((ma)<<8 | (mi))

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

1.5 Linux字符型驱动程序框架

- Linux设备驱动程序的代码结构大致可以分为如下几个部分：

  - 注册设备

    在系统初启，或者模块加载时候，必须将设备登记到相应的设备数组，并返回设备的主驱动号

  - 定义功能函数

    对于每一个驱动函数来说，都有一些和此设备密切相关的功能函数，那最常用的块设备或者字符设备来说，都存在着诸如open() read() write() ioctrol()这一类的操作。当系统社用这些调用时，将自动的使用驱动函数中特定的模块，来实现具体的操作。而对于特定的设备，上面的系统调用对应的函数是一定的。

  - 卸载设备

博芯电子

Procnip

Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

一个最简单字符驱动程序，由下面7个函数和1个结构体就可组成。Open（），Release,（）Write（），Read（）Ioctl（）Init（），Exit（）Struct file_operation

```
static int my_open(struct inode * inode, struct file * filp)
{ 设备打开时的操作 … }
```

```
static int my_release(struct inode * inode, struct file * filp)
{ 设备关闭时的操作 … }
```

```
static int my_write(struct file *file, const char * buffer, size_t count,
loff_t * ppos)
{ 设备写入时的操作 … }
```

```
static int my_read(struct file *file, const char * buffer, size_t count,
loff_t * ppos)
{ 设备读取时的操作 … }
```

博芯 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理 uml.org.cn

```
Static int my_ioctl(struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg)
{ 设备的控制操作…… }
```

```
static int __init my_init(void)
{初始化硬件，注册设备，创建设备节点… }
```

```
static void __exit my_exit(void)
{删除设备节点，注销设备… }
```

```
static struct file_operations my_fops = {
对文件操作结构体成员定义初始值…
}
```

博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

驱动接

struct file_operations

int (*seek) (struc

int (*read) (struc

int (*write) (str

int (*readdir) (s

int (*select) (st

int (*ioctl) (struc

int (*mmap) (str

int (*open) (str

int (*release) (st

int (*fsync) (stru

int (*fasync) (struc

int (*check_media_change) (struct inode * ,struct file *);

int (*revalidate) (dev_t dev);

}

用来修改一个文件的当前读写位置，并将新位

用来从设备中读取数据。函数返回一个非负值

向设备发送数据

对于设备节点来说，这个字段应该为NULL；它

用于程序询问设备是否可读和可写，或是否一

系统调用ioctl提供一中调用设备相关命令的

用来将设备内存映射到进程内存中

这总是操作在设备节点上的第一个操作，用于

当节点被关闭时调用这个操作

刷新设备

异步触发

只用于块设备，尤其是象软盘这类可移动介质。
内核调用这个方法判断设备中的物理介质（如

revalidate与缓冲区高速。缓存有关

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

文件操作结构体初始化

```
static struct file_operations my_fops =
{
    .owner = THIS_MODULE,
    .read  = sep4020_key_read,
    .write = sep4020_key_write,
    .ioctl = sep4020_key_ioctl,
    .open  = sep4020_key_open,
    .release = sep4020_key_release,
};
```

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

博芯电子
Prochip
Your Embedded Partner

2.基于Gpio的Linux字符型驱动设计

火龙果•整理
uml.org.cn

2.1 流水灯Linux驱动步骤

- 第一步： 编写字符设备驱动
- 第二步：加载
- 第三步：编写应用程序测试设备驱动

博芯电子 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理 uml.org.cn

2.2 第一步：编写流水灯Linux驱动

在/linux-3.2/driver/char/sep4020_char/下面新建一个sep4020_flowled.c

注意这个目录在后面会用到

- 内容如下：

```
#define KEY_MAJOR 249 /* 主设备号*/
#define LED_ON 1
#define LED_OFF 2
struct led_dev
{
    struct cdev cdev;
    unsigned char value;
};
struct led_dev *leddev;
```

博芯电子 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Reserved. Not to be reproduced by any means without prior written consent.

火龙果•整理 uml.org.cn

打开和关闭操作

- open和release函数会在设备打开和关闭时被调用，open的时候对设备进行初始化

```
static int sep4020_flowled _open(struct inode * inode, struct file * filp){
    sep4020_flowled_setup();
    ……
}
static int sep4020_flowled _release(struct inode * inode, struct file * filp
    ……
    return 0;
}
```

博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

写入和读出操作

```
static int sep4020_flowled _write(struct file *file, const char * buffer,
 size_t count, loff_t * ppos){
return 0;
}
```

```
static int sep4020_flowled _read(struct file *file, const char * buffer,
 size_t count, loff_t * ppos){
return 0;
}
```

博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

设备操作

```
int sep4020_flowled_ioctl(struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg)
{
    struct led_dev *dev = filp->private_data;
    unsigned int i,j;
    switch (cmd){
      case LED_ON:
        j=i%12;
        if(j<6)
          dev->value |=1<<j;
        else dev->value >>=1;
        *(volatile unsigned long*)GPIO_PORTE_DATA_V = dev->value; //flow led is open;
        i++;
        break;
      case LED_OFF:
        dev->value = 0;
        *(volatile unsigned long*)GPIO_PORTE_DATA_V = 0;            //flow led is close;
        break;
      default:
        return -ENOTTY;
    }
    return 0;
}
```

Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

设备初始化

```
static int __init sep4020_flowled _init(void){
        //申请设备号
        dev_t devno = MKDEV(KEY_MAJOR, 0);
        if(KEY_MAJOR)
                result = register_chrdev_region(devno, 1, "sep4020_flowled");
        /*动态申请设备结构体的内存*/
        leddev = kmalloc(sizeof(struct led_dev),GFP_KERNEL);
        if (!leddev)
        {
                result = -ENOMEM;
                goto fail_malloc;
        }
        //硬件初始化,推荐在open中实现
        //sep4020_flowled_setup();
        //字符设备注册
        cdev_init(&(leddev->cdev), &sep4020_flowled_fops);
        leddev->cdev.owner = THIS_MODULE;
        err = cdev_add(&leddev->cdev, devno, 1);// 创建设备文件
        return 0;
        fail_malloc: unregister_chrdev_region(devno,1);
        return result;
}
```

Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

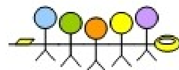
设备注销

```
static void __exit sep4020_flowled_exit(void)
{
        //删除设备文件
        cdev_del(&leddev->cdev);
        kfree(leddev);
        unregister_chrdev_region(MKDEV(KEY_MAJOR, 0),1);//注销设备
        unregister_chrdev(Led_Major, DEVICE_NAME);
}
```

```
module_init(sep4020_flowled _init); //向Linux系统记录设备初始化的函数名称
module_exit(sep4020_flowled_exit); //向Linux系统记录设备退出的函数名称
```

博芯电子 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.



修改Kconfig和Makefil

这两步要小心
容易出错！

- 在相应的字符型驱动的目录项
  句：

  config SEP4020_FLOWLED

  tristate "sep4020 flowed led"

- 在相同目录底下的Makefile中添加如下语句：

  obj-$(CONFIG_SEP4020_FLOWLED) +=
  sep4020_flowled.o

博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

第二步： 驱动程序的加载

- Linux内核有2种加载驱动程序的方法:
- 静态：

  Linux系统启动时,通过代码自身加载模块.这种方式称为静态编译入内核, 驱动程序开发完毕后一般使用这种方式.

- 动态：

  Linux系统启动后,通过insmod等命令加载模块.这种方式称为动态加载,驱动程序开发调试过程中一般使用这种方式.

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

方法1：驱动程序以驱动模块加载

- 打开终端，进入Linux根目录，输入命令make menuconfig
- 进入device drivers->character device->sep4020 char devices->sep4020 key driver
- 使用空格键将sep4020_flowed选择成M
- 运行make 命令,编译通过后当前目录下就生成名为sep4020_flowled.ko的驱动程序

博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

模块动态加载

```
#insmod sep4020_flowled.ko
```

- 驱动程序模块插入内核

```
#cat /proc/devices
```

- 查看是否载入,如果载入成功会显示你的设备名称sep4020_flowled

```
#rmmod sep4020_flowled.ko
```

- 从内核移除设备

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

动态加载在开发板端的操作：

- （1）将开发板上电，并将sep4020_flowled.ko拷贝到网络文件系统/demo/目录下面
- （2）在/dev/目录下创建一个设备节点flowled

  /dev # mknod flowled c 249 0
- （3）驱动程序模块插入内核

  insmod sep4020_flowled.ko
- （4）#cat /proc/devices

  查看是否载入,如果载入成功会显示你的设备名称sep4020_flowled

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

方法2：静态编译进内核

- 打开终端，进入Linux根目录，输入命令make menuconfig
- 进入device drivers->character device->sep4020 char devices->sep4020 key driver
- 使用空格键将sep4020_flowed选择成*
- 运行make 命令,编译通过后就将流水灯驱动编译进内核了
- 执行mkimage指令重新生成新的能被uboot引导的内核

博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

静态编译开发板端的操作：

- （1）将重新编译好的内核重新拷贝至tftp目录下，重新开发板上电

- （2）在/dev/目录下创建一个设备节点flowled

  # mknod /dev/flowled c 249 0

博芯电子 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理 uml.org.cn

第三步：编译应用程序

```
#include <stdio.h>
#define OPEN 1
#define CLOSE 2
int main(int argc， char
**argv)
{
 int fd;
 int i，j;
 fd = open("/dev/flowled"，0);
 if(fd == -1)
 {
   printf("wrong\r\n");
   exit(-1);
 }
 for(j=0; j<41; j++)
 {
 ioctl(fd， OPEN，0);
 for(i=0; i<1000000; i++);
 }
 close(fd);
 return 0;
}
```

新建一个文件**flowled.c**，注意应用程序和驱动的区分

博芯电子 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理 uml.org.cn

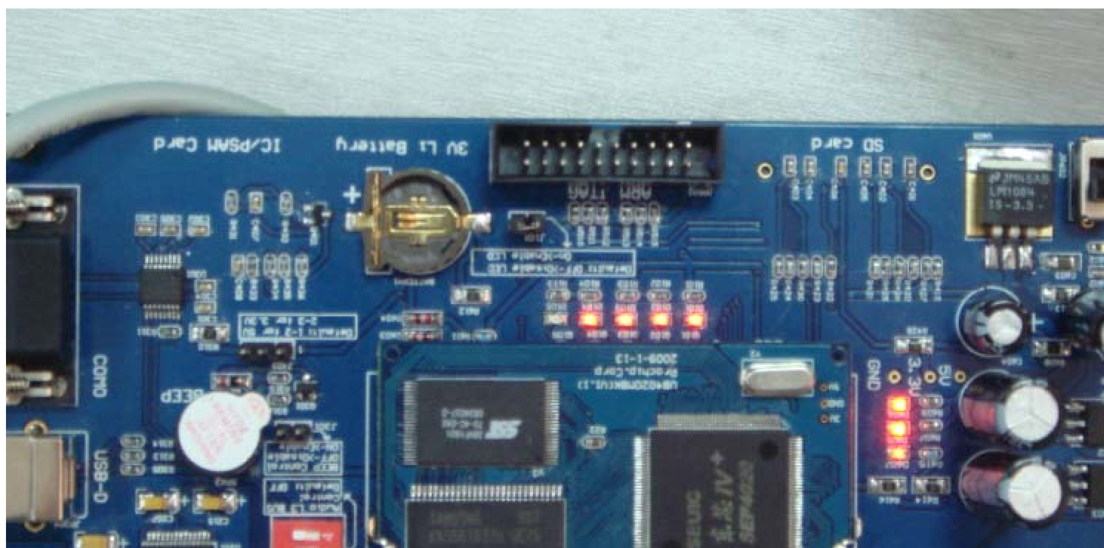
- 利用arm-linux-gcc将其编译为可执行的二进制文件：
- 指令如下：arm-linux-gcc –o flowled flowled.c
- 将编译好的flowled文件拷贝至nfs文件夹下

- 注意把led跳线帽插上。

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

流水灯演示

```
starting pid 221, tty '': '-/bin/sh'
/ # mknod /dev/flowled c 249 0
/ # insmod sep4020_flowled.ko
/ # ./flowled
```



博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

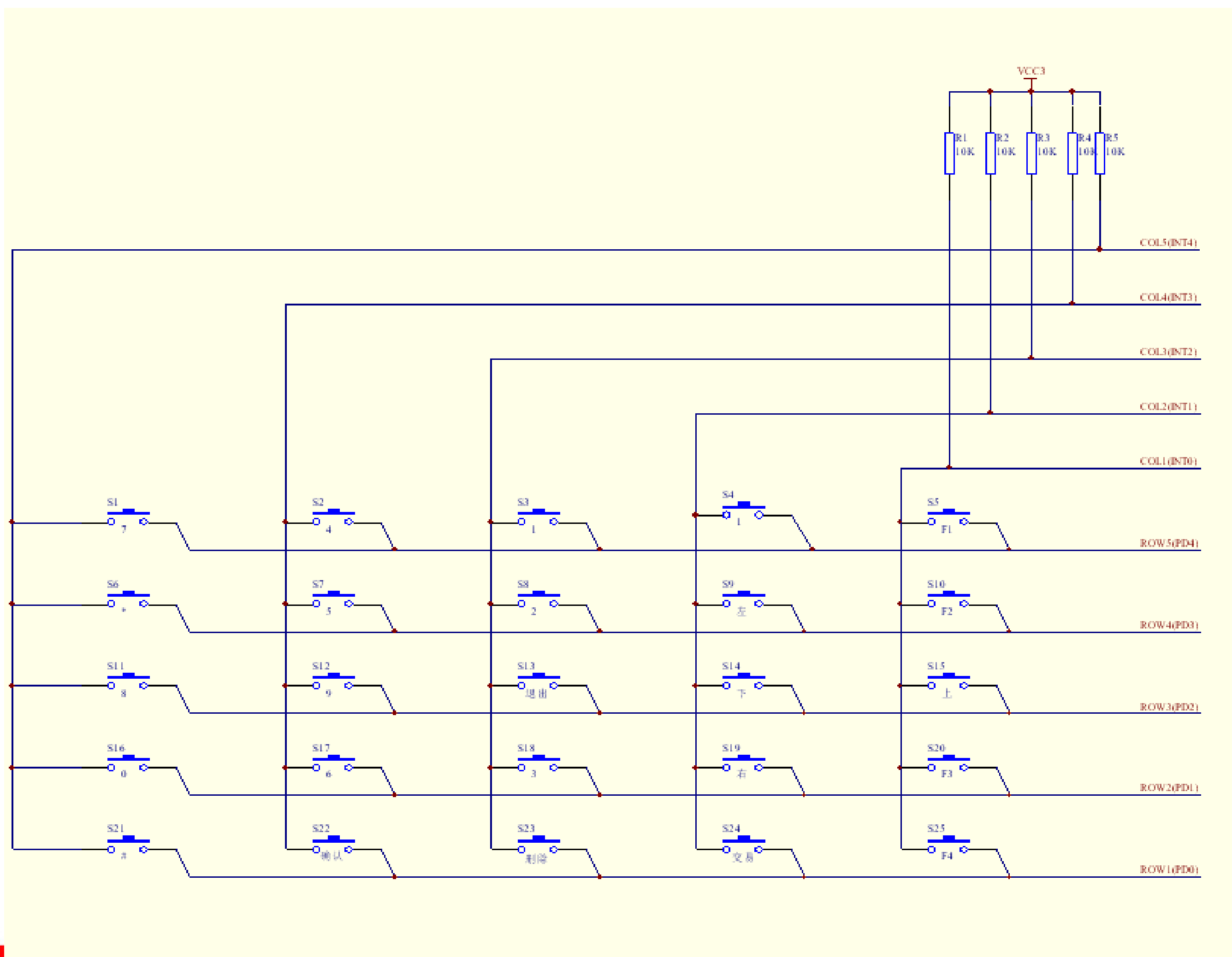
火龙果•整理
uml.org.cn

博芯电子
Prochip
Your Embedded Partner

3.Linux键盘驱动的设计

火龙果•整理
uml.org.cn

5X5键盘硬件原理图



博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果●整理
uml.org.cn

5X5键盘原理

- 列中断，行扫描

  在按下一个按键时，在列线会得到一个低电平的中断的信号，从而得到列线的数值，在产生中断后，在中断处理函数中对行线进行扫描，从而得到行线的数值，通过列线和行线确定按键的位置。

- 消除抖动：在判断有键按下后，进行一个软件的短延时（软件定时器），再判断键盘状态，如果仍处于按键按下状态，则可以判定该按键被按下，否则认为是一次抖动

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

Linux中断的使用：

安装中断处理程序

<linux/sched.h>

```
int request_irq(unsigned int irq, void (*handler)(int, void*, struct pt_regs *), unsigned long flags, const char *device, void *dev_id);

void free_irq(unsigned int irq, void *dev_id);
```

该参数为中断号

指向要安装的中断处理函数的指针

这个指针用于共享的中断信号线

这是一个与中断管理有关的各种选项的字节掩码
SA_INTERRUPT
SA_SHIRQ
SA_SAMPLE_RANDOM

用于显示中断的拥有者

江苏Linux公共技术服务中心

博心电子 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Reserved.
Not to be reproduced by any means without prior written consent.

火龙果•整理 uml.org.cn

Linux中断使用范例：

- 注册：

  request_irq(INTSRC_EXTINT0,sep4020_key_irqhandler,SA_INTERRUPT,"4020KEY",NULL)

- 中断处理函数：

  static irqreturn_t sep4020_key_irqhandler(int irq, void *dev_id, struct pt_regs *reg)

- 注销：

  free_irq(INTSRC_EXTINT0,NULL);

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

Linux定时器的使用：

- 1.定义一个定时器：

struct timer_list key_timer;

- 2.初始化定时器：

setup_timer(&key_timer,key_timer_handler,0);

- 3.定时器处理函数：

static void key_timer_handler(unsigned long arg)；

博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

Linux定时器的使用：

- 4.为定时器添加定时时间：

  key_timer.expires = jiffies + KEY_TIMER_DELAY_LONGTOUCH;

- 5.启动定时器：

  add_timer(&key_timer);

- 6.删除定时器：

  del_timer(struct timer_list *timer)

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

Init加载函数

```
/*注册中断函数*/
sep4020_request_irqs();
cdev_init(&(dev->cdev), &sep4020_key_fops);
dev->cdev.owner = THIS_MODULE;
setup_timer(&key_timer,key_timer_handler,0);
err = cdev_add(&dev->cdev, devno, 1);
if(err)
        printk("adding err\r\n");
return 0;
fail_malloc: unregister_chrdev_region(devno,1);
return result;
```

博芯电子 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理 uml.org.cn

Init函数中的中断申请函数

```
if(request_irq(INTSRC_EXTINT0,sep4020_key_irqhandler,SA_INTERRUPT,
    "4020KEY",NULL))    //申请中断
    goto  irq0_fail;
if(request_irq(INTSRC_EXTINT1,sep4020_key_irqhandler,SA_INTERRUPT,
    "4020KEY",NULL))
    goto  irq1_fail;
if(request_irq(INTSRC_EXTINT2,sep4020_key_irqhandler,SA_INTERRUPT,
    "4020KEY",NULL))
    goto  irq2_fail;
if(request_irq(INTSRC_EXTINT3,sep4020_key_irqhandler,SA_INTERRUPT,
    "4020KEY",NULL))
    goto  irq3_fail;
if(request_irq(INTSRC_EXTINT4,sep4020_key_irqhandler,SA_INTERRUPT,
    "4020KEY",NULL))
    goto  irq4_fail;
```

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

exit卸载函数

```
static void __exit sep4020_key_exit(void)
{
        sep4020_free_irqs();
        cdev_del(&dev->cdev);
        kfree(dev);
        unregister_chrdev_region(MKDEV(KEY_MAJOR, 0),1);
}
```

博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

Exit中的中断释放函数

- free_irq(INTSRC_EXTINT0,NULL);
- free_irq(INTSRC_EXTINT1,NULL);
- free_irq(INTSRC_EXTINT2,NULL);
- free_irq(INTSRC_EXTINT3,NULL);
- free_irq(INTSRC_EXTINT4,NULL);

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

Open函数

- int sep4020_key_open(struct inode *inode, struct file *filp)
- {

```
keystatus = KEY_UP;
sep4020_key_setup();
return 0;
```

- }

博芯电子 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理 uml.org.cn

Open中的setup函数

- 初始化所有用到的gpio口线

```
 maskkey();
*(volatile unsigned long*)GPIO_PORTD_SEL_V  |= 0x1F ;      //for common use
*(volatile unsigned long*)GPIO_PORTD_DIR_V  &= (~0x1F);    //0 stands for OUT
*(volatile unsigned long*)GPIO_PORTD_DATA_V &= (~0x1F);    //输出拉低

*(volatile unsigned long*)GPIO_PORTA_SEL_V |= 0x001F ;      //for common use
*(volatile unsigned long*)GPIO_PORTA_DIR_V |= 0x001F ;       //1 stands for in

*(volatile unsigned long*)GPIO_PORTA_INCTL_V |= 0x001F;     //中断输入方式
*(volatile unsigned long*)GPIO_PORTA_INTRCTL_V |= 0x03ff;   //中断类型为低电平解发
*(volatile unsigned long*)GPIO_PORTA_INTRCLR_V |= 0x001F;   //清除中断
*(volatile unsigned long*)GPIO_PORTA_INTRCLR_V = 0x0000;        //清除中断
unmaskkey();
```

博芯电子 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理 uml.org.cn

中断处理函数

- maskkey();
- *(volatile unsigned long*)GPIO_PORTA_INTRCLR_V |= 0x001F;
- *(volatile unsigned long*)GPIO_PORTA_INTRCLR_V = 0x0000;
- keystatus = KEY_UNSURE;
- key_timer.expires = jiffies + k
- add_timer(&key_timer);
- return IRQ_HANDLED;

什么时候**umaskkey**？

博芯电子 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理 uml.org.cn

Timer到期的中断处理函数

```
int j = 0;
j = *(volatile unsigned long*)GPIO_PORTA_DATA_V;
if ((j&0x1f) != 0x1f)
{
        if(keystatus == KEY_UNSURE)
        {
                keystatus = KEY_DOWN;
                key_timer.expires = jiffies + KEY_TIMER_DELAY2;
                keyevent();
                real_keynum = keynum;
                up(&key_sem);
                //printk("%s\n",KEY_MAP[real_keynum]);
                add_timer(&key_timer);

        }
        else
        {
                key_timer.expires
                add_timer(&key_
        }
}
else
{
        keystatus = KEY_UP;
        unmaskkey();
```

Here is unmaskkey !

博芯电子 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理 uml.org.cn

获取键值函数：keyevent

```
Init_Col = (*(volatile unsigned long*)(INTC_ISR_V)>>1) & 0x001F;
for( i = 0 ; i < 5 ; i ++)    //获取列数
{
        if( (Init_Col >> i) & 0x01 )
        {
                col = i ;
                col_count ++;
        }
}
if ((!read_col(col)))
{
        for(i=0;i<5;i++)            //通过对轮流PD0~PD4输出高电平；获取行数。
        {
        write_row(i,1);
        for(delay=0;delay<100;delay++); //delay
        if( read_col( col ) )
         {
                row = i ;
        row_count++;
         }
        write_row(i,0);
         }
}
if(col_count==1 && row_count==1)
{
        keynum = col*5 + row;

}
```

博芯电子 Prochip Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese[rved]
Not to be reproduced by any means without prior written consent.

火龙果•整理 uml.org.cn

Read函数

unsigned long err;

-
- /* 如果ev_press等于0，休眠 */
- down_interruptible(&key_sem);
- /* 将按键状态复制给用户，并清0 */
- printk("%d\n",KEY_MAP[real_keynum]);
- err = put_user(KEY_MAP[real_keynum],buf);
- return err ? -EFAULT : 0;

信号量?
内核空间，用户空间
?

博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

信号量

- 是用于保护临界区和同步的常用方法
- 与自旋锁不同，当获取不到信号量时，进程不会原地打转还是进入休眠等待状态

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

信号量的使用1：

- 1.定义：

  struct semaphore key_sem;

- 2.初始化信号量：

  void sema_init(struct semaphore *sem, int val)

  互斥信号量定义：

  void init_MUTEX(struct semaphore *sem)

  void init_MUTEX_LOCKED(struct semaphore *sem)

  快捷方式：DECLARE_MUTEX(name)；

  DECLARE_MUTEX_LOCKED(name);

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

信号量使用2：

- 3获得信号量：

  void down(struct semaphore *sem);

  使用该函数而进入睡眠状态的进程不可被信号打断；

  int down_interruptible(struct semaphore *sem);

  使用该函数而进入睡眠状态的进程可以被信号打断；
  在使用该函数时，一般对返回值进行检查，如果非0，立即返回

  If(down_interruptible(&sem))

  {return -error}

  int down_trylock(struct semaphore *sem);

  该函数尝试获得信号量sem，如果能立刻获得，就获得信号量并返回0，否则返回非0，它不会导致调用者睡眠

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

信号量的使用3：

- 释放信号量：

  void up(struct semaphore *sem)

  释放信号量sem，唤醒等待者

博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn

用户空间，内核空间

- Linux的虚拟地址空间也为0～4G。Linux内核将这4G字节的空间分为两部分。将最高的1G字节（从虚拟地址0xC0000000到0xFFFFFFFF），供内核使用，称为“内核空间”。而将较低的3G字节（从虚拟地址0x00000000到0xBFFFFFFF），供各个进程使用，称为“用户空间）。
- 因为每个进程可以通过系统调用进入内核，因此，Linux内核由系统内的所有进程共享。于是，从具体进程的角度来看，每个进程可以拥有4G字节的虚拟空间。

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

用户空间和内核空间的互访

- 由于内核空间和用户空间的内存不能互相访问，必须借助系统函数
- copy_from_user完成用户空间到内核空间的多字节复制
- copy_to_user完成内核空间到用户空间的多字节复制

- put_user完成内核空间到用户空间的简单类型复制
- get_user完成用户空间到内核空间的简单类型复制

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

验证键盘驱动：

- 将驱动加载至内核中（模块加载，静态加载）
- 编写相应的键盘应用程序，交叉编译，运行

```
int fd,i;
char buf[1];
fd = open("/dev/buttons",O_RDONLY);
if(fd == -1)
{
  printf("wrong\r\n");
  exit(-1);
}
for(i = 0; i < 1000; i++)
{
read(fd,buf,1);
printf("the key value is %d \n",buf[0]);
}
//printf("after open\n");
close(fd);

return 0;
```

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

如何设计一个优秀的键盘驱动？

- 缓冲区
- 对用户阻塞非阻塞读取的判断
- 参考sdk3.2 linux源码包中的键盘驱动

  位置：/linux/drivers/char/sep4020_char/sep4020_key.c

博芯电子
Prochip
Your Embedded Partner
Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.
火龙果•整理
uml.org.cn

The End!

Thanks!

博芯电子
Prochip
Your Embedded Partner

Copyright 2007 Prochip Electronics Co，ltd. All Rights Rese
Not to be reproduced by any means without prior written consent.

火龙果•整理
uml.org.cn