**驱动框架**

User space里的read，open，write怎样对应到driver驱动程序里的read，open，write：

驱动程序 drv\_init里用register\_chrdev(major,…,&drv\_fops)

结构体drv\_fops: ex. .read=led\_read

Led\_open, led\_read, led\_write

内核：VFS

Chrdev数组

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | ……. | major |  |  |  |  |  |

找到file\_operation

C库

User space:

Open(“/dev/xxx”), open, read

Char major

框架：

Struct file operations drv\_fops = {

.read = led\_read;

.owner =This\_Moudle;

.write = led\_write;

}

static int led\_open(struct inode \*inode, struct file \*file)

{设置硬件信息：寄存器清零并确定是输入或是输出

}

ssize\_t led\_read(struct file \*file, char \_\_user \*buf, size\_t count, loff\_t \*offp)

{从硬件寄存器读出数值并用copy\_to\_user(buf,key\_val,sizeof(key\_val)))

}

ssize\_t led\_write(struct file \*file, char \_\_user \*buf, size\_t count, loff\_t \*offp)

{用copy\_from\_user(&val, buf, count)把数据从user app读入，并把此值写入相应的寄存器;

}

static int \_\_init led\_init(void)

{ //“118”若为0，则自动配置major number

register\_chrdev(118, "led\_zxb", &led\_fops)；

用ioremap把寄存器的物理地址PA转为虚拟地址VA

}

static void \_\_exit led\_exit(void)

{unregister\_chrdev(118, "led\_zxb");//卸载register\_chrdev

iounmap(gpbcon);//对应ioremapap

}

//用module修饰一下led\_init，led\_exit使之成为入口和出口函数

module\_init(led\_init);

module\_exit(led\_exit);

MODULE\_LICENSE("GPL");

MODULE\_AUTHOR("zxb");

……….continuous in the future