**触摸屏操作**

**1、文件IO的其他应用**

系统IO除了可以访问LCD液晶屏设备进行颜色显示之后，还可以访问触摸屏获取用手点击的坐标。

**2、触摸屏设备文件**

在linux下，一切都是文件。连触摸屏都是文件。

设备名字：/dev/input/event0

**3、触摸屏获取坐标编程思路**

1）访问触摸屏设备文件

fd = open("/dev/input/event0");

2）直接读取该文件上的数据就可以。

read(fd);

3）分析读取出来的数据是什么类型？

int ? char ? char []? \*? 结构体

4）根据获取到的坐标做进一步的判断

if(坐标满足某一个条件)

{

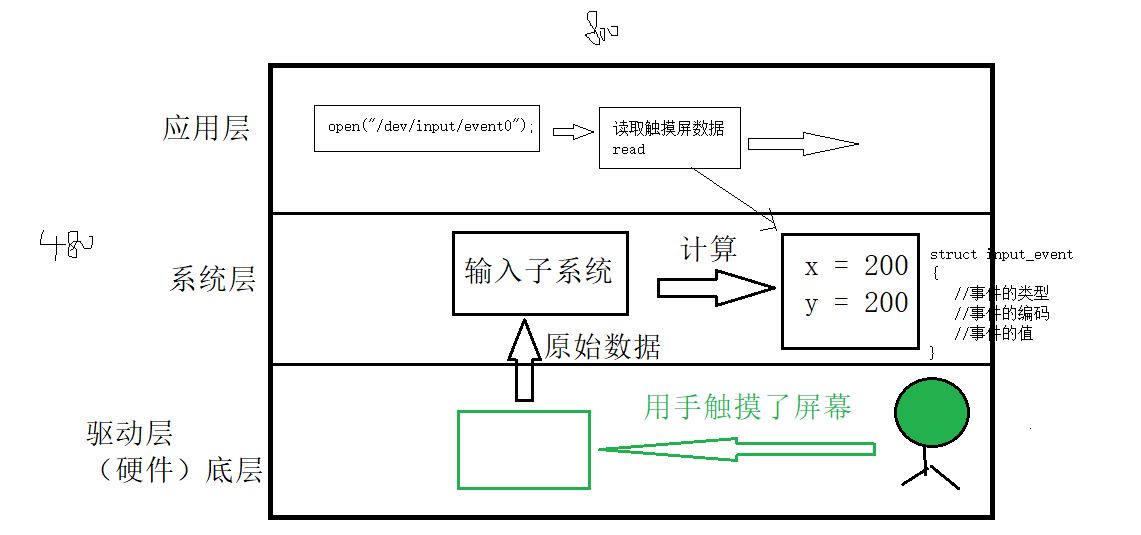
}

5）关闭文件。

close();

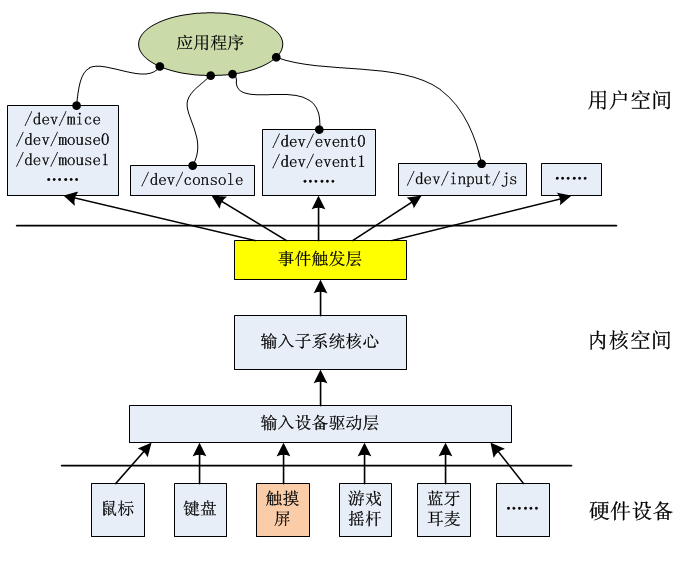
**4、研究触摸屏的数据类型**

如果想知道从文件event0中读取出来的数据是什么类型，首先必须先知道输入子系统计算完结果之后，这个结果是以什么方式存放在event0这个文件。



**5、输入子系统**

连接操作系统的输入设备，可不止一种，也许是一个标准PS/2键盘，也许是一个USB鼠标，或者是一块触摸屏，甚至是一个游戏机摇杆，Linux在处理这些纷繁各异的输入设备的时候，采用的办法还是找中间层来屏蔽各种细节，请看下图：



在Linux的内核中，对输入设备的使用，实际上运用了3大块来管理，他们分别是所谓的输入设备驱动层、输入子系统核心层，以及事件触发层。他们各自的工作分别是：

输入设备驱动层：（系统自动完成）

每一种设备都有其特定的驱动程序，他们被妥当地装载到操作系统的设备模型框架内，封装硬件所提供的功能，向上提供规定的接口。

核心层：（系统自动完成）

此处将收集由设备驱动层发来的数据，整合之后触发某一事件。

事件触发层：

这一层是我们需要关注的，我们可以通过在用户空间读取相应设备的节点文件来获知某设备的某一个动作。

以触摸屏为例，当手指在屏幕上滑动的时候，数据流大致是这样的：驱动层中的触摸屏驱动会源源不断地产生触摸屏相关数据，并向上递送给内核输入子系统，输入子系统进一步将这些信息规整为统一的结构体，并借助事件触发层发往对应的设备节点，至此，应用程序即可从这些设备节点读取相关信息。

值得注意的是，底层驱动产生的设备数据与上层应用读取设备数据是两个完全异步的过程，彼此之间是没有耦合和约束的，例如：当底层驱动产生的触摸屏坐标信息比应用层读取的速度要快时，应用程序将会丢失一部分坐标信息。

总结：

1）事件：当一些外接控制设备（鼠标、键盘、WIFI、触摸屏，按键）接入到嵌入式平台(GEC6818)时，这些外接设备的状态发生了改变(鼠标的左键被按下了，键盘的某一个按键被按下，有人连接到wifi上，有人摸了我的屏幕一下)时，这个动作就是事件。

2）输入子系统：当事件发生的时候，就是由输入子系统来计算这些事件中产生的值。

**6、输入信息结构体**

在最靠近应用程序的事件触发层上，内核所获知的各类输入事件，比如键盘被按了一下，触摸屏被滑了一下等，都将被统一封装在一个叫做 input\_event 的输入信息结构体当中，这个结构体定义如下：

/usr/include/linux/input.h

vincent@ubuntu:/usr/include/linux/$ cat input.h -n

1 #ifndef \_INPUT\_H

2 #define \_INPUT\_H

...

...

20

21 struct input\_event {

22 struct timeval time;//事件发生的时间

23 \_\_u16(int) type;//区分事件的类型（（鼠标/键盘/触摸屏）还是坐标） -> 鼠标/键盘/触摸屏

24 \_\_u16(int) code; //事件的编码（将坐标分为X和Y） -> 对事件进一步的描述 左键/A键/x轴

25 \_\_s32(int) value;//事件的值（触摸屏按下还是松开，坐标值）

26 };

27

...

分析结果：

在 /usr/include/linux/input-event-codes.h 中 定义了事件的类型、编码

结论： 当我们发生一次事件时，这个输入子系统就会返回一个值(结构体)给我们。

**7、分析结构体成员**

**1. time**

输入事件发生的时间戳，精确到微秒。时间结构体定义如下：

struct timeval

{

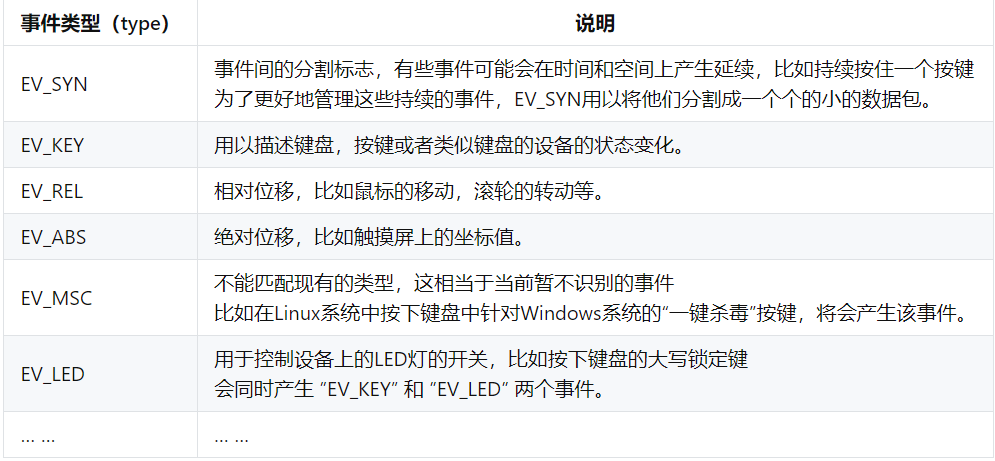
\_\_time\_t tv\_sec; // 秒

long int tv\_usec; // 微秒（1微秒 = 10-3毫秒 = 10-6秒）

};

**2. type**

输入事件的类型。比如：



EV\_KEY:按键（键盘，鼠标，触摸屏）事件

EV\_ABS:坐标事件

**注意：触摸屏上的按键事件和坐标事件是俩件事**

**3. code**

这个 **事件代码** 用于对事件的类型作进一步的描述。比如：当发生EV\_KEY事件时，则可能是键盘被按下了，那么究竟是哪个按键被按下了呢？此时查看code就知道了。当发生EV\_REL事件时，也许是鼠标动了，也许是滚轮动了。这时可以用code的值来加以区分。



当type=EV\_KEY&&code=BTN\_TOUCH:说明发生触摸屏按下松开事件

**4. value**

当code都不足以区分事件的性质的时候，可以用value来确认。比如由EV\_REL和REL\_WHEEL确认发生了鼠标滚轮的动作，但是究竟是向上滚还是向下滚呢？再比如由由EV\_KEY和KEY\_F确认了发生键盘上F键的动作，但究竟是按下呢还是弹起呢？这时都可以用value值来进一步判断。



代码：

#include <linux/input.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <fcntl.h>

#include <stdio.h>

int main()

{

    int touch\_fd = open("/dev/input/event0",O\_RDWR);

    if(touch\_fd == -1)

    {

        printf("打开触摸屏文件失败\n");

        return -1;

    }

    struct input\_event ts;

    while(1)

    {

        read(touch\_fd,&ts,sizeof(ts));

        printf("ts.type=%d\n",ts.type);

        printf("ts.code=%d\n",ts.code);

        printf("ts.value=%d\n",ts.value);

    }

close(touch\_fd);

return 0;

}

结果：

ts.type=3 --> EV\_ABS --> 坐标事件

ts.code=0 --> ABS\_X --> X轴事件

ts.value=355 --> X轴坐标

ts.type=3 --> EV\_ABS --> 坐标事件

ts.code=1 --> ABS\_Y --> Y轴事件

ts.value=218 --> Y轴坐标

ts.type=1 --> EV\_KEY --> 按键事件

ts.code=330 --> BTN\_TOUCH --> 触摸屏事件

ts.value=1 --> 按下事件

ts.type=1 --> EV\_KEY --> 按键事件

ts.code=330 --> BTN\_TOUCH --> 触摸屏事件

ts.value=0 --> 松开事件

注意：按下和松开事件的结构体成员的值都是固定的，X轴和Y轴的前2个变量也是固定的。

练习：

实现点击触摸屏显示坐标，每次点击触摸屏只显示一次坐标

printf("(%d %d)\n",x,y);

#include <linux/input.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <fcntl.h>

#include <stdio.h>

int main()

{

    int touch\_fd = open("/dev/input/event0",O\_RDWR);

    if(touch\_fd == -1)

    {

        printf("打开触摸屏失败\n");

        return -1;

    }

    struct input\_event ts;

    int x,y;

    while(1)

    {

        read(touch\_fd,&ts,sizeof(ts));

        // printf("ts.type = %d\n",ts.type);

        // printf("ts.code = %d\n",ts.code);

        // printf("ts.value = %d\n",ts.value);

        if(ts.type == 3 && ts.code == 0)//判断是否为X轴

            x = ts.value\*800/1022;//0~1022 0~800 \*800/1022:将1022坐标改成800坐标

        if(ts.type == 3 && ts.code == 1)//判断是否为Y轴

            y = ts.value\*480/598;//0~598 0~480 \*480/598：将598坐标改成480坐标

        if(ts.type == 1 && ts.code == 330 && ts.value == 1)//判断是否按下触摸屏

            printf("(%d %d)\n",x,y);

    }

}