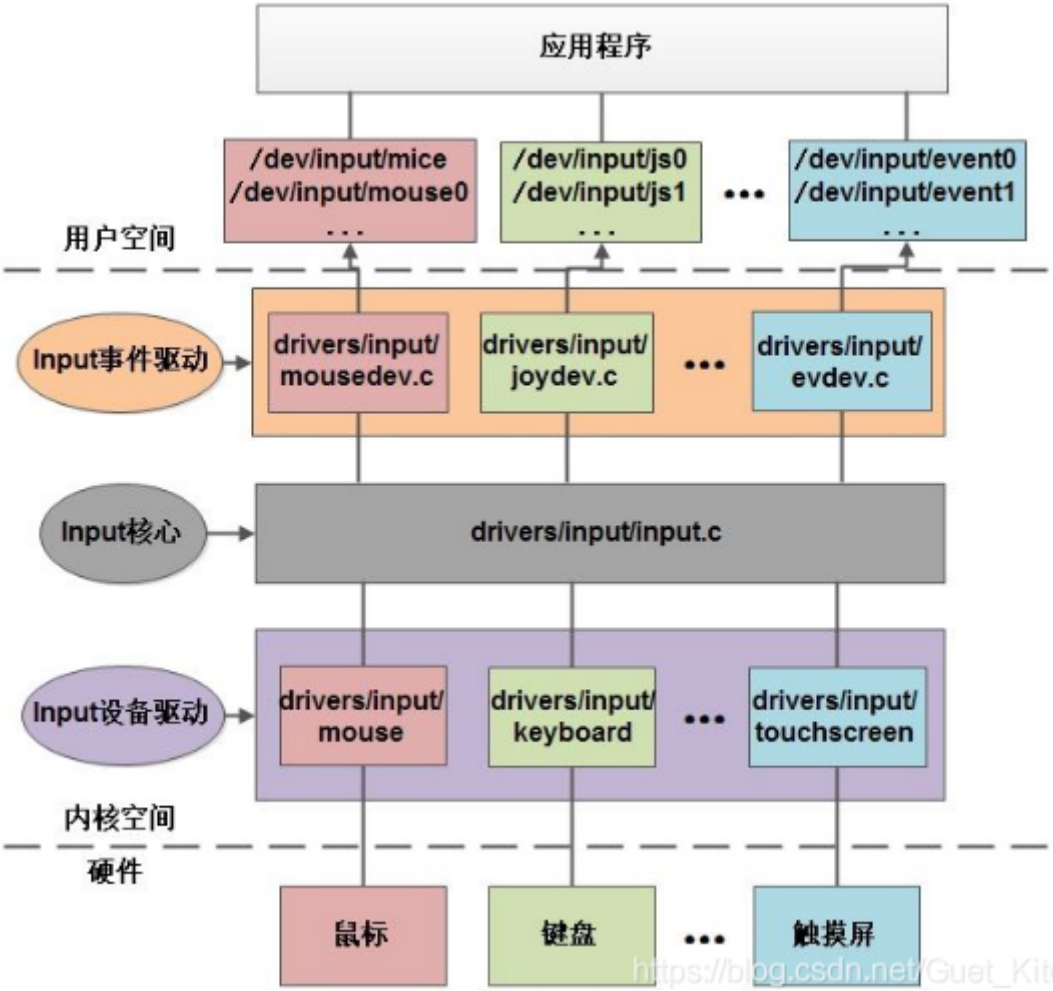
一、input子系统基本框架

Linux内核为了两个目的：

1. 简化纯输入类外设（如：键盘、鼠标、游戏杆、轨迹球、触摸屏。。。等等）的驱动开发
2. 统一输入类外设产生的数据格式（struct input_event），更加方便应用层编程

设计了输入子系统



事件处理层：接收来自核心层上报的事件，并选择对应的handler（事件处理器 struct input_handler）去处理。内核维护着多个事件处理器对象，每个input_handler对象专门处理一类事件，所有产生同类事件的设备驱动共用同一个handler。

设备驱动层：主要实现获取硬件设备的数据信息（包括触摸屏被按下、按下位置、鼠标移动、键盘按下等等），并转换为核心层定义的规范事件后提交给核心层，该层每个设备对应一个struct input_dev对象，

核心层：负责连接设备驱动层和事件处理层，为设备驱动层提供输入设备驱动的接口（struct input_dev）以及输入设备驱动的注册函数（input_register_device），为事件处理层提供输入事件驱动的接口；通知事件处理层对事件进行处理。

二、驱动开发步骤

```
/*init或probe函数中：
1. 创建struct input_dev对象input_allocate_device
2. 设置事件类型以及相关参数set_bit
3. 注册struct input_dev对象input_register_device
*/

/*exit或remove函数中：
1. 注销struct input_dev对象input_unregister_device
2. 销毁struct input_dev对象input_free_device
*/

/*上报事件
    两种事件上报方式：
    1. 对有中断支持的输入设备：在其中断处理函数（上半部或下半部）中上报事件
    2. 对无中断支持的输入设备：使用workqueue循环定时上报(struct delayed_work)
    主要函数：
    input_event
    input_report_abs
    input_sync
*/
```

相关接口：

```
/*_init*/
struct input_dev *input_allocate_device(void)//创建对象

void set_bit(struct input_dev *dev,unsigned long whichbits)//设置事件类型

void input_set_abs_params(struct input_dev *dev,unsigned int axis,int min,int
max,int fuzz,int flat)

int input_register_device(struct input_dev *dev)//注册input设备到内核

/*_exit*/
void input_unregister_device(struct input_dev *dev)
void input_free_device(struct input_dev *dev)

/*上报事件*/
void input_event(struct input_dev *,unsigned int t,unsigned int c,int v)

void input_report_key(struct input_dev *,unsigned int c,int v) //上报按键事件
void input_report_abs(struct input_dev *,unsigned int c,int v)//上报绝对坐标事件

void input_sync(struct input_dev *)//上报完成后需要调用这些函数来通知系统处理完整事件

/*应用层数据类型*/
struct input_event {
    struct timeval time;        // 时间戳
    __u16 type;                 // 事件类型
    __u16 code;                 // 哪个分值
```

```
    __s32 value;            // 具体值
};
```

三、key2-input版代码解析

四、mpu6050-input版代码解析