# 浅析linux下鼠标驱动的实现

转载 2009年10月16日 13:02:00 标签：linux / input / report / struct / buffer / list / 2204

zpf1217

原创 38 粉丝 7 喜欢 0 评论 34

等级： 博客 4 访问量： 13万+

积分： 1846 排名： 2万+

对于鼠标驱动和前面分析过的键盘驱动都是共用input模型,所以,对于事件上报和处理的方式都没有区别,只是mouse鼠标驱动当上报完dx,dy,left,middle,right之后,需要调用input_sync(),将前面上报的仅仅填充在缓冲区中的数据,通过mousedev_notify_readers()发送给open了的挂接在mousedev->client_list链表上等待获取鼠标信息的client门,鼠标设备和键盘设备类似都是在/dev/input/目录下创建了一个char类型的设备节点,由应用程序使用read或者poll来阻塞调用,对于键盘设备为/dev/input/event0,...,/dev/input/eventx,对于鼠标设备为/dev/input/mouse0,...,/dev/input/mousex,可以使用sudo cat /dev/input/event0来从终端上截获显示按键的信息,使用sudo cat /dev/input/mouse0来捕捉鼠标的信息.

让我们来看看驱动源码【gliethttp.Leith】:

==========drivers/input/mouse/amimouse.c==========

```
    input_report_rel(amimouse_dev, REL_X, dx);
    input_report_rel(amimouse_dev, REL_Y, dy);

    input_report_key(amimouse_dev, BTN_LEFT, ciaa.pra & 0x40);
    input_report_key(amimouse_dev, BTN_MIDDLE, potgor & 0x0100);
    input_report_key(amimouse_dev, BTN_RIGHT, potgor & 0x0400);

    input_sync(amimouse_dev);// 拷贝到open了的每个client的client->packets[16]环形缓冲区,每个应用程序在调用open 时,mousedev_open都会调用kzalloc来申请一个独立的mousedev_client结构体,然后将该client挂接到 mousedev->client_list链表,最后由mousedev_notify_readers向mousedev->client_list链表上挂接的每个client拷贝鼠标信息,最后wake_up唤醒read或poll.
```

==========drivers/input/mousedev.c==========

mousedev_read=>mousedev_packet=>如果dx,dy,dz同时都为0,说明鼠标停止了,那么client->ready = 0;

mousedev_event(dev, EV_SYN, SYN_REPORT, 0)=>mousedev_notify_readers=>如果dx,dy,dz有一个发生了移动或者鼠标按键上一次的按键不同,那么client->ready = 1;拷贝数据到mousedev->client_list链表上挂接的每个client的环形缓冲区,最后调用wake_up_interruptible(&mousedev->wait);唤醒因为read或者poll操作而被pending住的应用程序,比如xWindows系统或者MiniGUI系统.

mousedev_write=>mousedev_generate_response=>向client->ps2[6]缓冲区填充数据,有效数据的个数为client->bufsiz,之后执行如下赋值client->buffer = client->bufsiz;让client->buffer等于client->ps2[6]数据缓冲区中有效数据的个数.

```
static ssize_t mousedev_read(struct file *file, char __user *buffer,
             size_t count, loff_t *ppos)
{
    struct mousedev_client *client = file->private_data;
    struct mousedev *mousedev = client->mousedev;
    signed char data[sizeof(client->ps2)];
    int retval = 0;

    if (!client->ready && !client->buffer && mousedev->exist &&
     (file->f_flags & O_NONBLOCK))
        return -EAGAIN;

    retval = wait_event_interruptible(mousedev->wait,
```

```
            !mousedev->exist || client->ready || client->buffer);
    //等待条件满足或者信号发生,client->ready和client->buffer都可以在调用wake_up_interru
ptible(&mousedev->wait)之后,因为为真,而继续往下执行.
    if (retval)
        return retval;

    if (!mousedev->exist)
        return -ENODEV;

    spin_lock_irq(&client->packet_lock);//禁止中断

    if (!client->buffer && client->ready) {
        mousedev_packet(client, client->ps2);
        client->buffer = client->bufsiz;
    }

    if (count > client->buffer)
        count = client->buffer;

    memcpy(data, client->ps2 + client->bufsiz - client->buffer, count);
//所以从这里可以看出,client->bufsiz为ps2[]数组有效数据索引的上限值,
//client->buffer为ps2[]数组索引的下限值
    client->buffer -= count;//这样之后,再次执行read时,将会接续该buffer偏移位置继续读取.

    spin_unlock_irq(&client->packet_lock);//打开中断

    if (copy_to_user(buffer, data, count))//拷贝到用户空间
        return -EFAULT;

    return count;
}
```

对于mouse和keyboard来说poll方法是同时处理多项输入的相当高效的信息处理方法,应用程序可以使用select或者poll甚至epoll来等待多个事件的发生,比如同时等待mouse和key的发生,然后来统一处理【gliethttp.Leith】.

```
static unsigned int mousedev_poll(struct file *file, poll_table *wait)
{
    struct mousedev_client *client = file->private_data;
    struct mousedev *mousedev = client->mousedev;

    poll_wait(file, &mousedev->wait, wait);
    return ((client->ready || client->buffer) ? (POLLIN | POLLRDNORM) : 0) |
        (mousedev->exist ? 0 : (POLLHUP | POLLERR));
}
```

以上鼠标input事件和键盘的input时间基本一致,最后都是调用input_report_rel()、input_report_key()等,不同的是mousedev_event只有当调用input_sync才会发生向client的数据拷贝动作,而键盘的evdev_event的事件处理函数不管是什么信息都会执行如下遍历:

```
list_for_each_entry_rcu(client, &evdev->client_list, node)
    evdev_pass_event(client, &event);
```

来完成向每个client数据buffer拷贝数据【gliethttp.Leith】.

阅读全文

本文已收录于以下专栏：

### Linux USB 鼠标驱动程序详解 这个好

USB 总线引出两个重要的链表！ 一个 USB 总线引出两个重要的链表，一个为 USB 设备链表，一个为 USB 驱动链表。设备链表包含各种系统中的USB 设备以及这些设备的所有接口，驱...

u012075739 2014年04月17日 16:58 2272

### Linux USB 鼠标驱动程序详解

USB 总线引出两个重要的链表！ 一个 USB 总线引出两个重要的链表，一个为 USB 设备链表，一个为 USB 驱动链表。设备链表包含各种系统中的USB 设备以及这些设备的所有接口，驱动链表包...

lidaqiang99 2011年07月10日 18:53 2106



### Linux USB 驱动开发实例（二）—— USB 鼠标驱动注解及测试

参考2.6.14版本中的driver/usb/input/usbmouse.c。鼠标驱动可分为几个部分：驱动加载部分、probe部分、open部分、urb回调函数处理部分。 一、驱动加载部分st...

zqixiao_09 2016年03月28日 20:05 3783

### Linux驱动之usb鼠标

应用程序获取鼠标数据输入参考：http://blog.csdn.net/qq_21792169/article/details/50809605 /* * 参考drivers\hid\usbhi...

qq_21792169 2015年09月28日 21:31 3767

### Linux USB 鼠标驱动程序解析

SB 总线引出两个重要的链表！ 一个 USB 总线引出两个重要的链表，一个为 USB 设备链表，一个为 USB 驱动链表。设备链表包含各种系统中的 USB 设备以及这些设备的所有接口，驱动链表包...

chinaunixj 2012年04月18日 12:48 499

### linux驱动之usb鼠标按键的读取

上一篇博文只是usb总线驱动程序的框架，下面来真正写一个usb驱动程序。 USB鼠标驱动，鼠标输入HID类型，其数据传输采用中断URB，鼠标端点类型为IN 目的：usb鼠标按键的驱动代码编写：...

a912293097 2015年01月01日 18:07 1721

### Linux USB 鼠标驱动程序解析

USB 总线引出两个重要的链表！ 一个 USB 总线引出两个重要的链表，一个为 USB 设备链表，一个为 USB 驱动链表。设备链表包含各种系统中的 USB 设备以及这些设备的所有接口，驱动...

xiaozhudedede 2017年03月22日 18:10 313

### Linux下的USB总线驱动（二）鼠标驱动分析

版权所有，转载请说明转自 http://my.csdn.net/weiqing1981127 2.USB鼠标驱动 usbmouse.c 下面我们分析下USB鼠标驱动，鼠标输入HID类型...


dragon101788 2013年07月17日 17:25 2219