对uio device的读取，取得的是该device的event number，也就是该device发生的interrupt number

in drivers/uio/uio.c

uio_read()就用于相应对uio device的读取

```
1.  static ssize_t uio_read(struct file *filep, char __user *buf,
2.                         size_t count, loff_t *ppos)
3.  {
4.          struct uio_listener *listener = filep->private_data;
5.          struct uio_device *idev = listener->dev;
6.          DECLARE_WAITQUEUE(wait, current);
7.          ssize_t retval;
8.          s32 event_count;
9.
10.         if (!idev->info->irq)
11.                 return -EIO;
12.
13.         if (count != sizeof(s32))                          ①
14.                 return -EINVAL;
15.
16.         add_wait_queue(&idev->wait, &wait);
17.
18.         do {
19.                 set_current_state(TASK_INTERRUPTIBLE);
20.
21.                 event_count = atomic_read(&idev->event);
22.                 if (event_count != listener->event_count) {        ②
23.                         if (copy_to_user(buf, &event_count, count))        ③
24.                                 retval = -EFAULT;
25.                         else {
26.                                 listener->event_count = event_count;

27.                                 retval = count;
28.                         }
29.                         break;                ④
30.                 }
```

```
31.
32.                 if (filep->f_flags & O_NONBLOCK) {            ⑤
33.                         retval = -EAGAIN;
34.                         break;
35.                 }
36.
37.                 if (signal_pending(current)) {    ⑥
38.                         retval = -ERESTARTSYS;
39.                         break;
40.                 }
41.                 schedule();                          ⑦
42.         } while (1);
43.
44.         __set_current_state(TASK_RUNNING);
45.         remove_wait_queue(&idev->wait, &wait);
46.
47.         return retval;
48. }
```

①

只能读取interrupt number，sizeof(s32)足够

②

只有uio device的event number != listener->event_count时才会返回

uio device的event number每次interrupt则+1

这是uio device的interrupt handler

```
static irqreturn_t uio_interrupt(int irq, void *dev_id)
{
        struct uio_device *idev = (struct uio_device *)dev_id;
        irqreturn_t ret = idev->info->handler(irq, idev->info);

        if (ret == IRQ_HANDLED)
                uio_event_notify(idev->info);

        return ret;
}
```

```
void uio_event_notify(struct uio_info *info)
{
        struct uio_device *idev = info->uio_dev;

        atomic_inc(&idev->event);
        wake_up_interruptible(&idev->wait);
        kill_fasync(&idev->async_queue, SIGIO, POLL_IN);
}
```

③

换回uio device的event number给user application

④

即当read uio device时，该device已经有interrupt发生，则理解返回

⑤

如果read uio device时,该device并没有interrupt发生，同时是非阻塞的read，则以EAGAIN返回

⑥

没有interrupt发生，则发起read的process本来应该睡眠，但如果此时有signal pending（即有signal发生，比如按了Ctrl+Z等）

则也不会阻塞，但需要返回ERESTARTSYS，通知user application返回的原因。

⑦

没有interrupt则阻塞。

user application可以通过发起阻塞read()来在user modeapplication中response device interrupt!

for example:

```
while(read(hdev, &int, 4) == 4)
{
    interrupr handling
}
```

对uio device的write只用于enable / disable 该device的interrupt。

in uiolib.c

```
1.  int uio_int_enable(uio_dev_t *hdev)
2.  {
3.      uio_dev_t *dev = hdev;
4.      int32_t irq_enable = 1;
5.
6.      if (dev == NULL)
7.          return -1;
8.
9.      if (dev->fd == -1)
10.         return -2;
11.
12.     if (!(dev->event_flags & UIO_EPOLL_FLAG_EVENT_ATTACHED))
13.         return -3;
14.
15.     if (write(dev->fd, &irq_enable, 4) != 4)
16.         return -4;
17.
18.     DBG_DEBUG("Enabled interrupt on device %s\n", dev->name);
19.
20.     return 0;
21. }
22.
23. int uio_int_disable(uio_dev_t *hdev)
24. {
25.     uio_dev_t *dev = hdev;
26.     int32_t irq_disable = 0;
27.
28.     if (dev == NULL)
29.         return -1;
30.
31.     if (dev->fd == -1)
```

```
32.             return -2;
33.
34.         if (!(dev->event_flags & UIO_EPOLL_FLAG_EVENT_ATTACHED))
35.             return -3;
36.
37.         if (write(dev->fd, &irq_disable, 4) != 4)
38.             return -4;
39.
40.         DBG_DEBUG("Disabled interrupt on device %s\n", dev->name);
41.
42.         return 0;
43.     }
```

in drivers/uio/uio.c

uio_write()就用于相应对uio device的write operation。

```
1.  static ssize_t uio_write(struct file *filep, const char __user *buf,
2.                          size_t count, loff_t *ppos)
3.  {
4.          struct uio_listener *listener = filep->private_data;
5.          struct uio_device *idev = listener->dev;
6.          ssize_t retval;
7.          s32 irq_on;
8.
9.          if (!idev->info->irq)
10.                 return -EIO;
11.
12.         if (count != sizeof(s32))
13.                 return -EINVAL;
14.
15.         if (!idev->info->irqcontrol)
16.                 return -ENOSYS;
17.
18.         if (copy_from_user(&irq_on, buf, count))
19.                 return -EFAULT;
20.
21.         retval = idev->info->irqcontrol(idev->info, irq_on);        ①
22.
23.         return retval ? retval : sizeof(s32);
24. }
```

①

write operation转换成struct uio_info的irqcontrol()调用。

对uio_pdrv_genirq 类型的uio device而言

in drivers/uio/uio_pdrv_genirq.c

```
1.  static int uio_pdrv_genirq_probe(struct platform_device *pdev)
2.  {
3.  ......
4.      uioinfo->irqcontrol = uio_pdrv_genirq_irqcontrol;
5.  ......
6.  }
```

```
static int uio_pdrv_genirq_irqcontrol(struct uio_info *dev_info, s32 irq_on)
{
        struct uio_pdrv_genirq_platdata *priv = dev_info->priv;
        unsigned long flags;

        /* Allow user space to enable and disable the interrupt
         * in the interrupt controller, but keep track of the
         * state to prevent per-irq depth damage.
         *
         * Serialize this operation to support multiple tasks and concurrency
         * with irq handler on SMP systems.
         */

        spin_lock_irqsave(&priv->lock, flags);
        if (irq_on) {
                if (__test_and_clear_bit(UIO_IRQ_DISABLED, &priv->flags))
                        enable_irq(dev_info->irq);
        } else {
                if (!__test_and_set_bit(UIO_IRQ_DISABLED, &priv->flags))
                        disable_irq_nosync(dev_info->irq);
        }
        spin_unlock_irqrestore(&priv->lock, flags);

        return 0;
}
```

uio_pdrv_genirq driver的write operation只支持对device interrupt的enable / disable action。