Kernel driver prog. day 6

Presented by

Hans de Goede

Wait Queues

- In a driver you may want to wait for a certain event, typically some io / hardware operation to finish
- The Linux kernel has a mechanism for this called wait queues
- Wait queues are typically used in the implementation of blocking read / write calls and the implementation of the poll call

Wait Queue example (1)

```
#include <linux/wait.h>
struct driver_data {
    spinlock_t lock;
    wait_queue_head_t wq;
    my_fifo_t fifo;
}
```

Wait Queue example (2)

```
bool driver_data_ready(struct driver_data *d) {
    bool ret;
    spin_lock(&d→lock);
    ret = !myfifo_empty(&d→fifo);
    spin_unlock(&d→lock);
    return ret;
}
```

Wait Queue example (3)

```
static ssize_t dev_read(struct file *f, char __user
    *data, size_t count, loff_t *ppos) {
  struct driver_data *d = f→private_data;
  int r, len; unsigned long flags;
  while (1) {
    spin_lock_irqsave(&d→lock, flags);
    if (!myfifo_empty(&d→fifo))
        break; /* We've data ready to read */
    spin_unlock_irqrestore(&d→lock, flags);
```

Wait Queue example (4)

```
if (f→flags & O_NONBLOCK)
    return -EAGAIN;
r = wait_event_interruptible_timeout(&d→wq,
        driver_data_ready(d), msecs_to_jiffies(5000));
if (r < 0)
    return r;
if (r == 0)
    return -EIO; /* timeout */
}
```

Wait Queue example (5)

```
    len = min(fifo_len(&d→fifo), count);
    r = fifo_copy_to_user(&d→fifo, data, len);
    spin_unlock_irqrestore(&d→lock, flags);
    return r;
}
```

Wait Queue example (6)

```
irq_return_t driver_irq(int irq, void *dev_id) {
    struct driver_data *d = dev_id;
    int i, len = read_data_avail_len(d);
    spin_lock(&d→lock);
    for (i = 0; i < len; i++)
        myfifo_add_data(&d→fifo, read_data(d));
    spin_unlock(&d→lock);
    wake_up_interruptible(&d->wq);
    return IRQ_HANDLED;
```

Wait Queue example (7)

```
int driver_probe(...) {
    ...
    init_waitqueue_head(&d→wq);
    ...
}
```

Questions?

Contact:
hdegoede@redhat.com

Hands on: Coding time!

Contact:
hdegoede@redhat.com