Poll and Select

Advanced Embedded Software Development

with **Dan Walkes**

University of Colorado **Boulder**

Learning objectives:

Understand use of poll and select in kernel drivers.

Understand seeking and associated read/write/open function implementation.

poll and select

- poll and select determine whether a device/file can be read without blocking or wait for a file descriptor to become ready
- Support implemented by poll method in device driver

```
struct file_operations {

    __poll_t (*poll) (struct file *, struct poll_table_struct *);
```

```
/*
 * The file operations for the pipe device
 * (some are overlayed with bare scull)
 */
struct file_operations scull_pipe_fops = {
        .owner =        THIS_MODULE,
        .llseek =       no_llseek,
        .read =         scull_p_read,
        .write =        scull_p_write,
        .poll =         scull_p_poll,
        .unlocked_ioctl = scull_ioctl,
        .open =         scull_p_open,
        .release =      scull_p_release,
        .fasync =       scull_p_fasync,
};
```

Linux Device Drivers 3rd Edition Chapter 6
https://elixir.bootlin.com/linux/v5.3.1/source/include/linux/fs.h#L1789
https://github.com/cu-ecen-5013/ldd3/blob/latest-lecture-source/scull/pipe.c
http://man7.org/linux/man-pages/man2/poll.2.html
http://man7.org/linux/man-pages/man2/select.2.html

poll and select

```
static unsigned int scull_p_poll(struct file *filp, poll_table *wait)
{
        struct scull_pipe *dev = filp->private_data;
        unsigned int mask = 0;

        /*
         * The buffer is circular; it is considered full
         * if "wp" is right behind "rp" and empty if the
         * two are equal.
         */
        mutex_lock(&dev->lock);
        poll_wait(filp, &dev->inq,  wait);
        poll_wait(filp, &dev->outq, wait);
        if (dev->rp != dev->wp)
                mask |= POLLIN | POLLRDNORM;    /* readable */
        if (spacefree(dev))
                mask |= POLLOUT | POLLWRNORM;   /* writable */
        mutex_unlock(&dev->lock);
        return mask;
}
```

```
struct scull_pipe {
        wait_queue_head_t inq, outq;       /* read and write queues */
```

- Call poll_wait on any wait queues which could indicate poll status changes
  - Kernel will wait on these as necessary
- Return bit mask describing currently available operations (read or write)

Linux Device Drivers 3rd Edition Chapter 6
https://github.com/cu-ecen-5013/ldd3/blob/latest-lecture-source/scull/pipe.c#L231

Read Rules

- When data is available in the input buffer:
  - read should return immediately with at least 1 byte
  - poll should return POLLIN|POLLRDNORM
- When no data in the input buffer
  - read
    - should block until one byte is there OR
    - if O_NONBLOCK return with value of -EAGAIN
  - poll should report unreadable (read flags all 0)

Linux Device Drivers 3rd Edition Chapter 6

Read Rules

- At end of file
  - read should return immediately with value of 0
  - poll should report POLLHUP

Write Rules

- When space is available in the output buffer:
  - write should return without delay, accepting at least one byte
  - poll should report POLLOUT | POLLWRNORM
- When the output buffer is full
  - write
    - should block until space is freed or
    - if O_NONBLOCK is set return -EAGAIN
  - poll should report file is not writable (write flags all 0)

Write Rules

- Never make a write call wait for data transmission (transfer from output buffer to device), even if O_NONBLOCK is not set.
  - To ensure transmissions complete to the device, the driver must provide fsync

Linux Device Drivers 3rd Edition Chapter 6

Seeking on a Device

- Default lseek just sets filp->f_pos
- llseek file operation can be implemented if the seek a custom operation for the device.
- Call nonseekable_open in your open function if seek doesn't make sense for your device
  - data flow rather than data area
    - serial port
    - keyboard

Linux Device Drivers 3rd Edition Chapter 6
https://github.com/cu-ecen-5013/ldd3/blob/latest-lecture-source/scull/pipe.c#L225

Access Control from open()

- Single-Open
  - Only one process can open at a time
  - scullsingle example - Obtain atomic in open, release in release()
- Single User
  - Compare process uid in open(), return -EBUSY when in use
- Alternative to returning -EBUSY - block in open()

Linux Device Drivers 3rd Edition Chapter 6