Kernel Sleep Continued

Advanced Embedded Software Development

with **Dan Walkes**

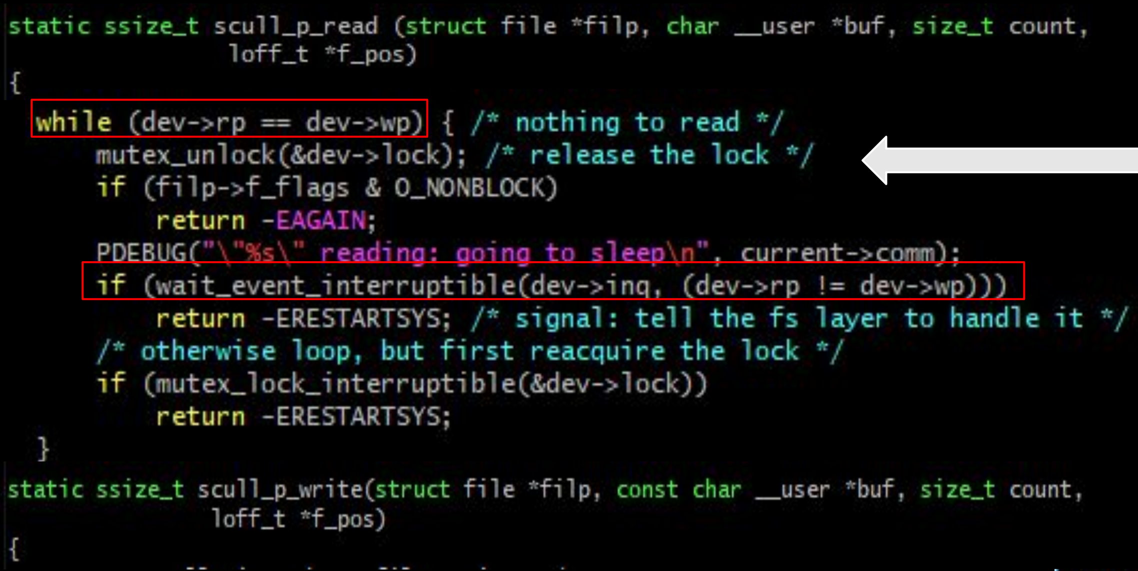
University of Colorado **Boulder**

Learning objectives:

Understand scull pipe wakeup kernel example from driver sources.

Understand nonblocking I/O implementation as an alternative to sleep.

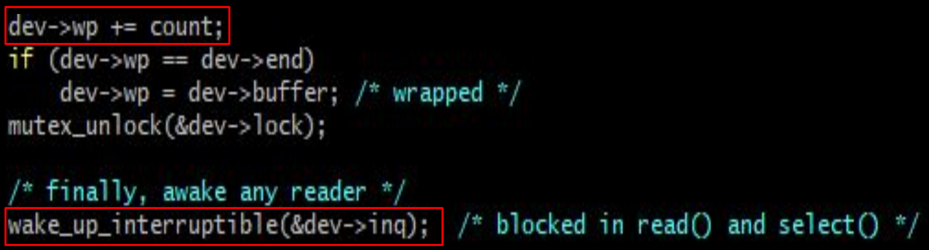
Scull Pipe Sleep/Wakeup Example



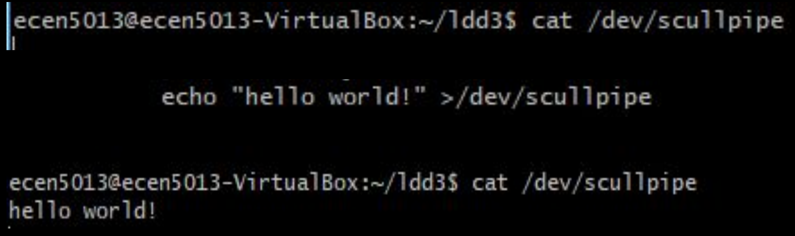
```
static ssize_t scull_p_read (struct file *filp, char __user *buf, size_t count,
                loff_t *f_pos)
{
  while (dev->rp == dev->wp) { /* nothing to read */
      mutex_unlock(&dev->lock); /* release the lock */
      if (filp->f_flags & O_NONBLOCK)
          return -EAGAIN;
      PDEBUG("\"%s\" reading: going to sleep\n", current->comm);
      if (wait_event_interruptible(dev->inq, (dev->rp != dev->wp)))
          return -ERESTARTSYS; /* signal: tell the fs layer to handle it */
      /* otherwise loop, but first reacquire the lock */
      if (mutex_lock_interruptible(&dev->lock))
          return -ERESTARTSYS;
  }
static ssize_t scull_p_write(struct file *filp, const char __user *buf, size_t count,
                loff_t *f_pos)
{
```

Why release the lock here?

- Don't sleep with lock held



```
dev->wp += count;
if (dev->wp == dev->end)
    dev->wp = dev->buffer; /* wrapped */
mutex_unlock(&dev->lock);

/* finally, awake any reader */
wake_up_interruptible(&dev->inq); /* blocked in read() and select() */
```



```
ecen5013@ecen5013-VirtualBox:~/ldd3$ cat /dev/scullpipe


            echo "hello world!" >/dev/scullpipe


ecen5013@ecen5013-VirtualBox:~/ldd3$ cat /dev/scullpipe
hello world!
```

https://github.com/cu-ecen-5013/ldd3/blob/master/scull/pipe.c#L121

Blocking I/O Buffering

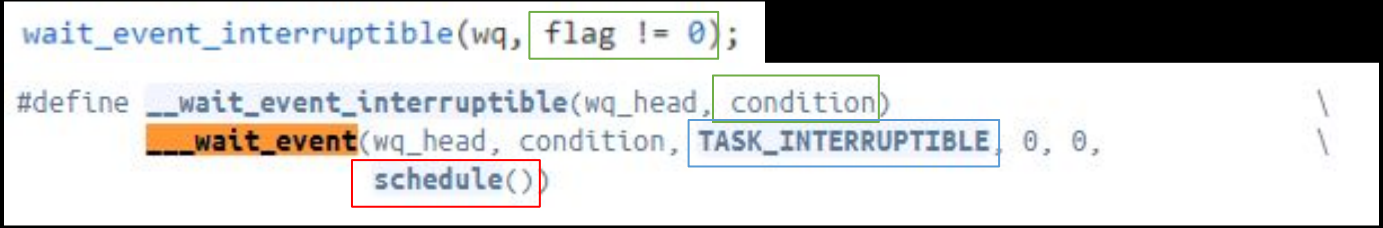
- Output and Input buffers are often useful for handling blocking I/O on real devices.
    - This means any blocking is on access to the buffer rather than access to the device
    - Especially large performance benefit when device interaction is much slower than memory access

How a process sleeps

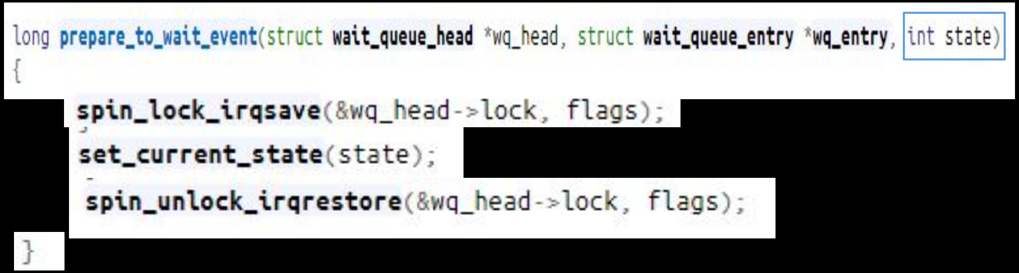
- Task states:
  - TASK_RUNNING - able to run, may not be executing
  - TASK_INTERRUPTIBLE and TASK_UNINTERRUPTIBLE - two types of sleep
- schedule()
  - function which runs the scheduler, allows other process to be scheduled (effectively puts the process to sleep)

Linux Device Drivers 3rd Edition Chapter 6
https://elixir.bootlin.com/linux/v5.3.1/source/include/linux/sched.h#L74
https://elixir.bootlin.com/linux/v5.3.1/source/include/linux/wait.h#L455

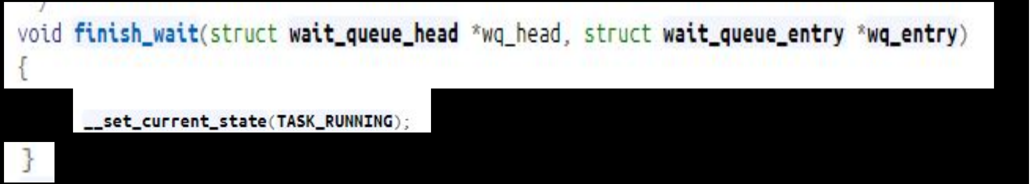
How a process sleeps



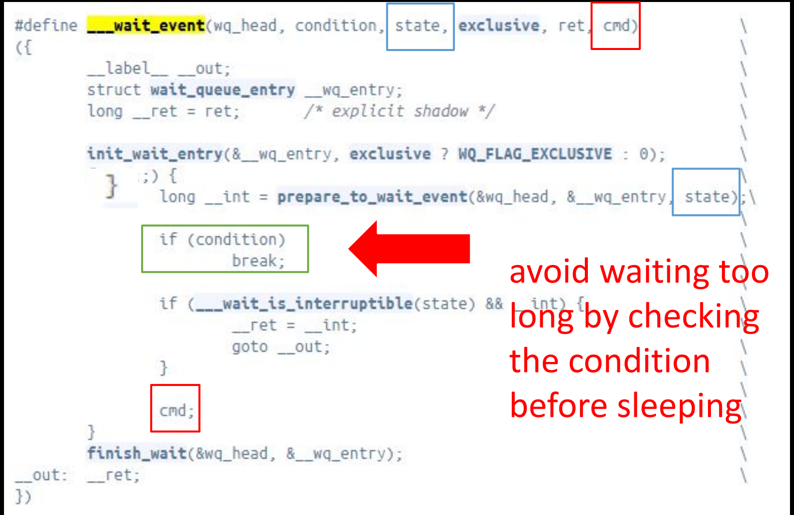
```
wait_event_interruptible(wq, flag != 0);

#define __wait_event_interruptible(wq_head, condition)                    \
        ___wait_event(wq_head, condition, TASK_INTERRUPTIBLE, 0, 0,       \
                      schedule())
```



```
long prepare_to_wait_event(struct wait_queue_head *wq_head, struct wait_queue_entry *wq_entry, int state)
{
        spin_lock_irqsave(&wq_head->lock, flags);
        set_current_state(state);
        spin_unlock_irqrestore(&wq_head->lock, flags);
}
```



```
void finish_wait(struct wait_queue_head *wq_head, struct wait_queue_entry *wq_entry)
{
        __set_current_state(TASK_RUNNING);
}
```



```
#define ___wait_event(wq_head, condition, state, exclusive, ret, cmd)          \
({                                                                              \
        __label__ __out;                                                        \
        struct wait_queue_entry __wq_entry;                                     \
        long __ret = ret;       /* explicit shadow */                           \
                                                                                \
        init_wait_entry(&__wq_entry, exclusive ? WQ_FLAG_EXCLUSIVE : 0);        \
         }    ;;) {                                                             \
                long __int = prepare_to_wait_event(&wq_head, &__wq_entry, state);\
                                                                                \
                if (condition)                                                  \
                        break;                                                  \
                                                                                \
                if (___wait_is_interruptible(state) && __int) {                 \
                        __ret = __int;                                          \
                        goto __out;                                             \
                }                                                               \
                                                                                \
                cmd;                                                            \
        }                                                                       \
        finish_wait(&wq_head, &__wq_entry);                                     \
__out:  __ret;                                                                  \
})
```

avoid waiting too long by checking the condition before sleeping

Linux Device Drivers 3rd Edition Chapter 6

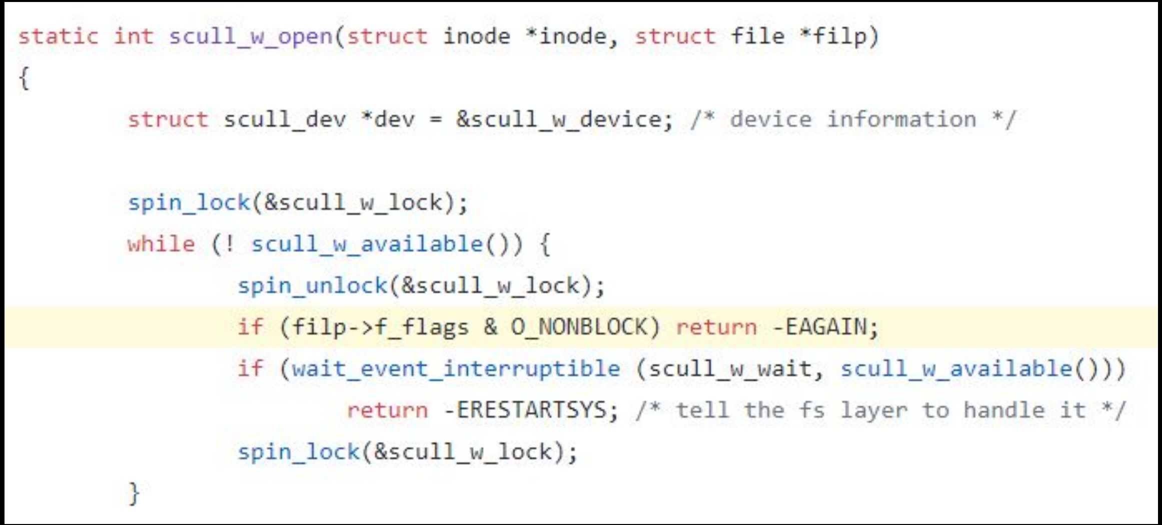
https://elixir.bootlin.com/linux/v5.3.1/source/include/linux/sched.h#L74

https://elixir.bootlin.com/linux/v5.3.1/source/include/linux/wait.h#L455

https://elixir.bootlin.com/linux/v5.3.1/source/kernel/sched/wait.c#L258

Nonblocking I/O

- What if process requests O_NONBLOCK on open()?
  - O_NONBLOCK will be set in filp->f_flags

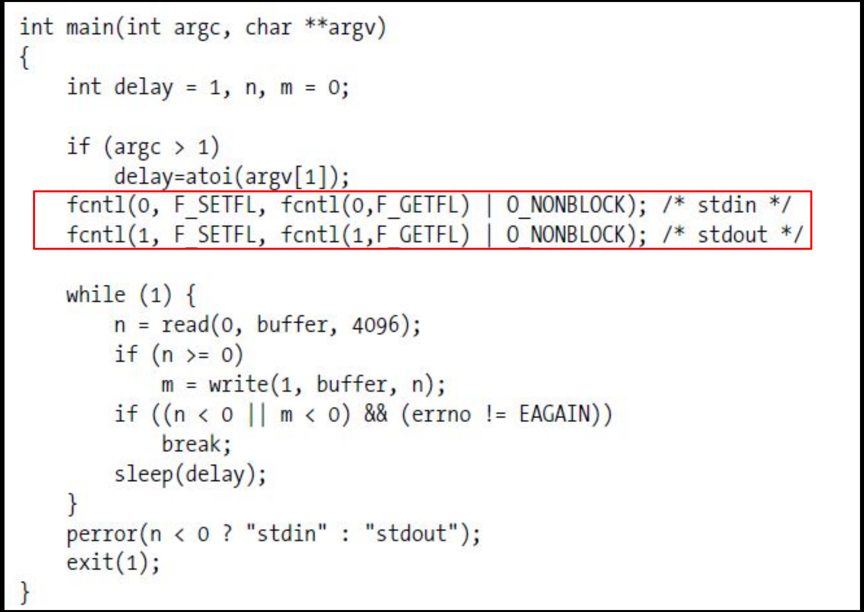


```
static int scull_w_open(struct inode *inode, struct file *filp)
{
        struct scull_dev *dev = &scull_w_device; /* device information */

        spin_lock(&scull_w_lock);
        while (! scull_w_available()) {
                spin_unlock(&scull_w_lock);
                if (filp->f_flags & O_NONBLOCK) return -EAGAIN;
                if (wait_event_interruptible (scull_w_wait, scull_w_available()))
                        return -ERESTARTSYS; /* tell the fs layer to handle it */
                spin_lock(&scull_w_lock);
        }
```

Linux System Programing Chapter 2
Linux Device Drivers 3rd Edition Chapter 6
https://github.com/cu-ecen-5013/ldd3/blob/latest-lecture-source/scull/access.c#L181

Testing Non-blocking IO

- Book references a nbtest example which demonstrates non-blocking operation

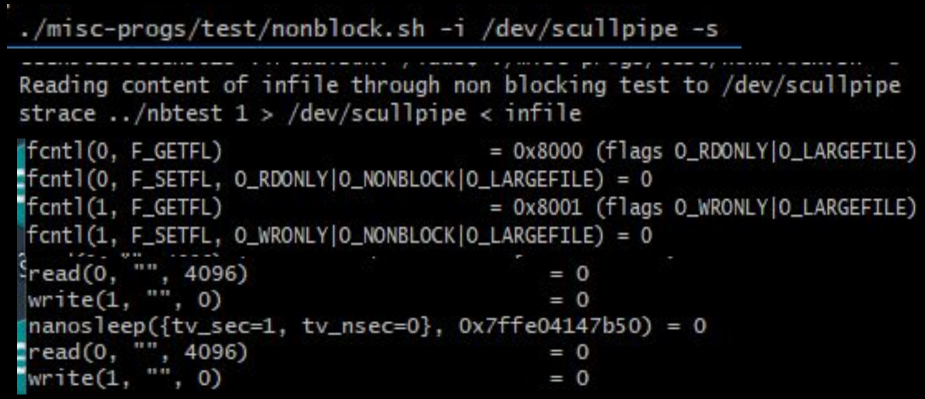


```
int main(int argc, char **argv)
{
    int delay = 1, n, m = 0;

    if (argc > 1)
        delay=atoi(argv[1]);
    fcntl(0, F_SETFL, fcntl(0,F_GETFL) | O_NONBLOCK); /* stdin */
    fcntl(1, F_SETFL, fcntl(1,F_GETFL) | O_NONBLOCK); /* stdout */

    while (1) {
        n = read(0, buffer, 4096);
        if (n >= 0)
            m = write(1, buffer, n);
        if ((n < 0 || m < 0) && (errno != EAGAIN))
            break;
        sleep(delay);
    }
    perror(n < 0 ? "stdin" : "stdout");
    exit(1);
}
```

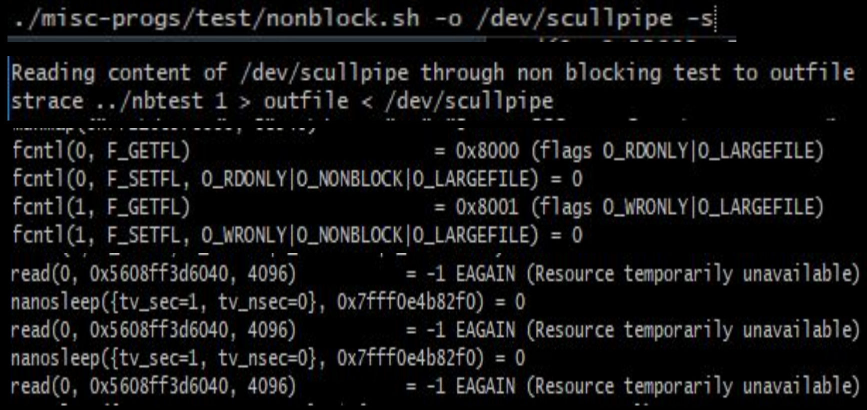
Linux Device Drivers 3rd Edition Chapter 6
https://github.com/cu-ecen-5013/ldd3/blob/latest-lecture-source/misc-progs/test/nonblock.sh

Testing Nonblocking IO

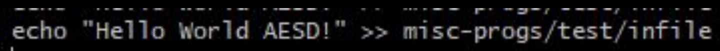
- Use misc-progs and nonblock.sh test script after loading the driver you want to test



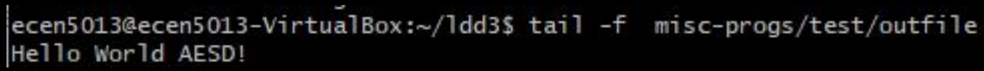
```
./misc-progs/test/nonblock.sh -i /dev/scullpipe -s

Reading content of infile through non blocking test to /dev/scullpipe
strace ../nbtest 1 > /dev/scullpipe < infile

fcntl(0, F_GETFL)                       = 0x8000 (flags O_RDONLY|O_LARGEFILE)
fcntl(0, F_SETFL, O_RDONLY|O_NONBLOCK|O_LARGEFILE) = 0
fcntl(1, F_GETFL)                       = 0x8001 (flags O_WRONLY|O_LARGEFILE)
fcntl(1, F_SETFL, O_WRONLY|O_NONBLOCK|O_LARGEFILE) = 0
read(0, "", 4096)                       = 0
write(1, "", 0)                         = 0
nanosleep({tv_sec=1, tv_nsec=0}, 0x7ffe04147b50) = 0
read(0, "", 4096)                       = 0
write(1, "", 0)                         = 0
```



```
./misc-progs/test/nonblock.sh -o /dev/scullpipe -s

Reading content of /dev/scullpipe through non blocking test to outfile
strace ../nbtest 1 > outfile < /dev/scullpipe

fcntl(0, F_GETFL)                       = 0x8000 (flags O_RDONLY|O_LARGEFILE)
fcntl(0, F_SETFL, O_RDONLY|O_NONBLOCK|O_LARGEFILE) = 0
fcntl(1, F_GETFL)                       = 0x8001 (flags O_WRONLY|O_LARGEFILE)
fcntl(1, F_SETFL, O_WRONLY|O_NONBLOCK|O_LARGEFILE) = 0
read(0, 0x5608ff3d6040, 4096)          = -1 EAGAIN (Resource temporarily unavailable)
nanosleep({tv_sec=1, tv_nsec=0}, 0x7fff0e4b82f0) = 0
read(0, 0x5608ff3d6040, 4096)          = -1 EAGAIN (Resource temporarily unavailable)
nanosleep({tv_sec=1, tv_nsec=0}, 0x7fff0e4b82f0) = 0
read(0, 0x5608ff3d6040, 4096)          = -1 EAGAIN (Resource temporarily unavailable)
```



```
echo "Hello World AESD!" >> misc-progs/test/infile
```

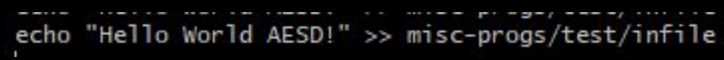


```
ecen5013@ecen5013-VirtualBox:~/ldd3$ tail -f  misc-progs/test/outfile
Hello World AESD!
```

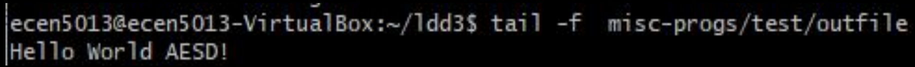
Linux Device Drivers 3rd Edition Chapter 6
https://github.com/cu-ecen-5013/ldd3/blob/latest-lecture-source/misc-progs/test/nonblock.sh
https://github.com/cu-ecen-5013/ldd3/blob/latest-lecture-source/misc-progs/nbtest.c

Testing Non-blocking IO

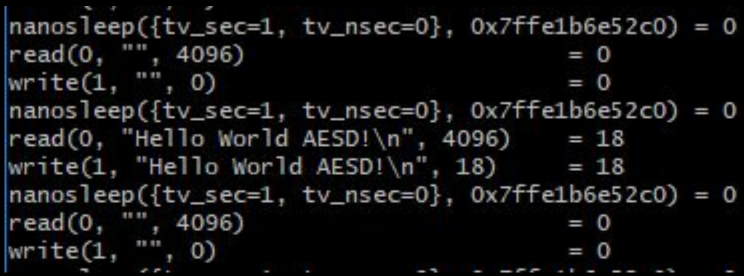
- Writing to misc-progs/test/infile results in output on misc-progs/test/outfile through non-blocking pipe



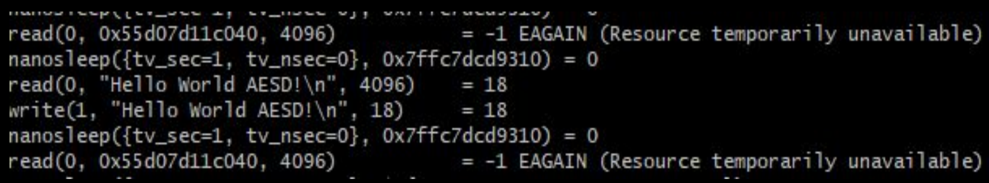
```
echo "Hello World AESD!" >> misc-progs/test/infile
```



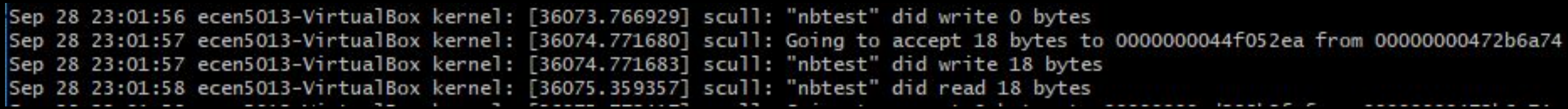
```
ecen5013@ecen5013-VirtualBox:~/ldd3$ tail -f  misc-progs/test/outfile
Hello World AESD!
```



```
nanosleep({tv_sec=1, tv_nsec=0}, 0x7ffe1b6e52c0) = 0
read(0, "", 4096)                       = 0
write(1, "", 0)                         = 0
nanosleep({tv_sec=1, tv_nsec=0}, 0x7ffe1b6e52c0) = 0
read(0, "Hello World AESD!\n", 4096)    = 18
write(1, "Hello World AESD!\n", 18)     = 18
nanosleep({tv_sec=1, tv_nsec=0}, 0x7ffe1b6e52c0) = 0
read(0, "", 4096)                       = 0
write(1, "", 0)                         = 0
```



```
read(0, 0x55d07d11c040, 4096)           = -1 EAGAIN (Resource temporarily unavailable)
nanosleep({tv_sec=1, tv_nsec=0}, 0x7ffc7dcd9310) = 0
read(0, "Hello World AESD!\n", 4096)    = 18
write(1, "Hello World AESD!\n", 18)     = 18
nanosleep({tv_sec=1, tv_nsec=0}, 0x7ffc7dcd9310) = 0
read(0, 0x55d07d11c040, 4096)           = -1 EAGAIN (Resource temporarily unavailable)
```



```
Sep 28 23:01:56 ecen5013-VirtualBox kernel: [36073.766929] scull: "nbtest" did write 0 bytes
Sep 28 23:01:57 ecen5013-VirtualBox kernel: [36074.771680] scull: Going to accept 18 bytes to 0000000044f052ea from 00000000472b6a74
Sep 28 23:01:57 ecen5013-VirtualBox kernel: [36074.771683] scull: "nbtest" did write 18 bytes
Sep 28 23:01:58 ecen5013-VirtualBox kernel: [36075.359357] scull: "nbtest" did read 18 bytes
```

Linux Device Drivers 3rd Edition Chapter 6
https://github.com/cu-ecen-5013/ldd3/blob/latest-lecture-source/misc-progs/test/nonblock.sh