DEVICE DRIVERS

BLG413E – System Programming, Practice Session 3

scull (Simple Character Utility for Loading Localities)

- a char driver that treats a memory area as a device
- used as an example to demonstrate and test the interface between the kernel and char drivers

Compiling scull

- **Warning:** simplified scull code under ninova is incompatible with Linux kernel versions newer than 2.6.35.
    - Use "*uname -a*" to check the version of the kernel you are currently using
- Two changes are made in the code to adapt to newer versions of the Linux kernel.

ioctl is not available after Linux kernel 2.6.36, use *unlocked_ioctl* instead

```
struct file_operations scull_fops = {
    .owner =     THIS_MODULE,
    .llseek =    scull_llseek,
    .read =      scull_read,
    .write =     scull_write,
    .unlocked_ioctl =  scull_ioctl,
    .open =      scull_open,
    .release =   scull_release,
};
```

init_MUTEX is not available after Linux kernel 2.6.37, use *sema_init* instead

```
/* Initialize each device. */
for (i = 0; i < scull_nr_devs; i++) {
    dev = &scull_devices[i];
    dev->quantum = scull_quantum;
    dev->qset = scull_qset;
    sema_init(&dev->sem,1);
    devno = MKDEV(scull_major, scull_minor + i);
    cdev_init(&dev->cdev, &scull_fops);
    dev->cdev.owner = THIS_MODULE;
    dev->cdev.ops = &scull_fops;
    err = cdev_add(&dev->cdev, devno, 1);
    if (err)
        printk(KERN_NOTICE "Error %d adding scull%d", err, i);
}
```

Compiling scull

- **Makefile:**

M=$(PWD) is to build external module in the working directory

```
obj-m := scull.o
all:
        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
```

- **Compiling:**

make

Using scull

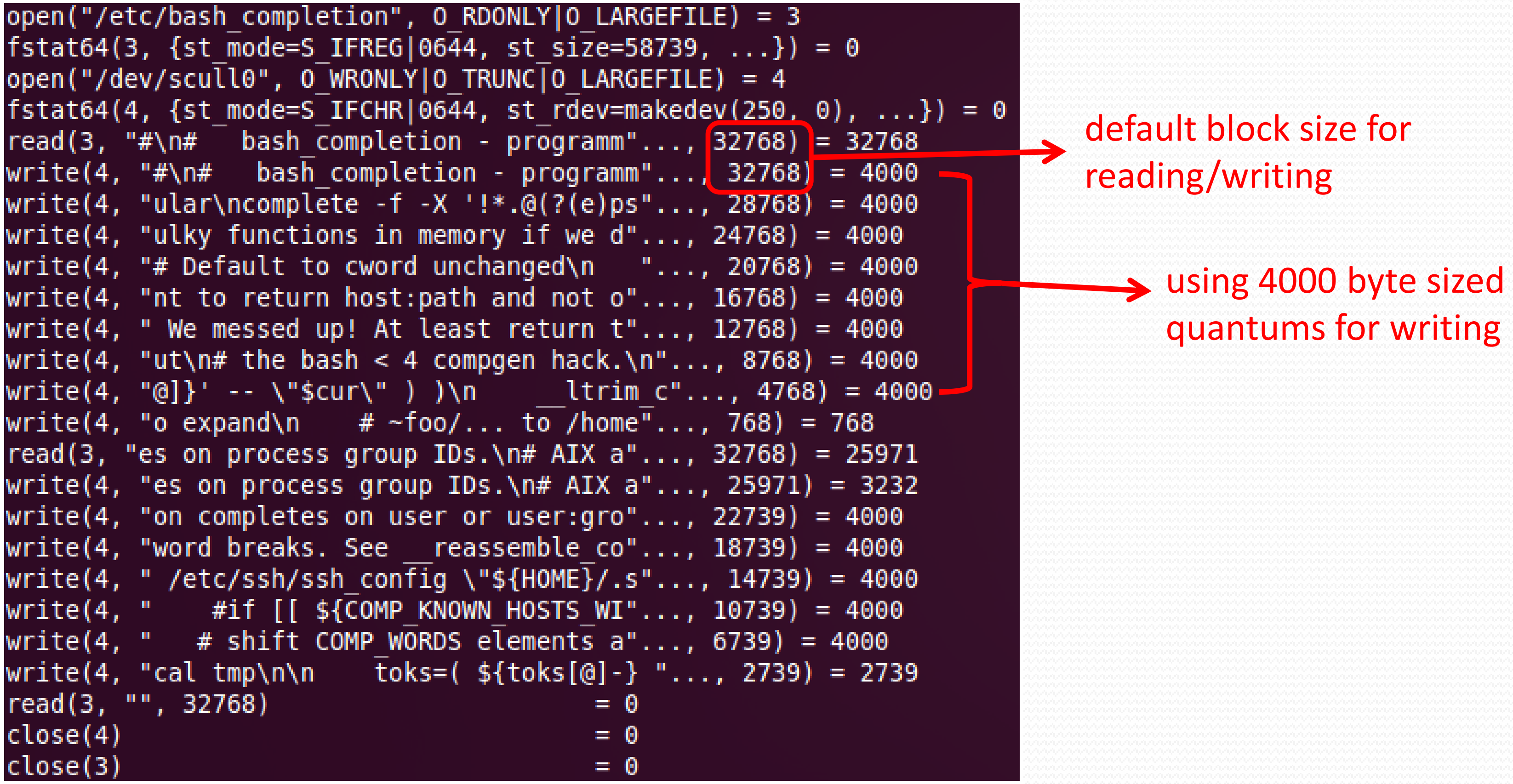
- when testing the scull module, it's better to become root instead of using sudo for commands:
  - *sudo su*
- **Loading:**
  - *insmod ./scull.ko*
  - *lsmod* → to see scull in the list of loaded modules
- **Getting the major number:**
  - *grep scull $\boxed{\text{/proc/devices}}$* → file displaying currently configured (and loaded) character and block devices
- **Creating the device nodes (assuming major number is 250):**
  - *mknod /dev/scull0 c 250 0*
  - *mknod /dev/scull1 c 250 1*
  - …

Using scull

- **<u>Writing to the device:</u>**
  - *echo testing > /dev/scull0*
- **<u>Reading from the device:</u>**
  - *cat /dev/scull0*

Using scull

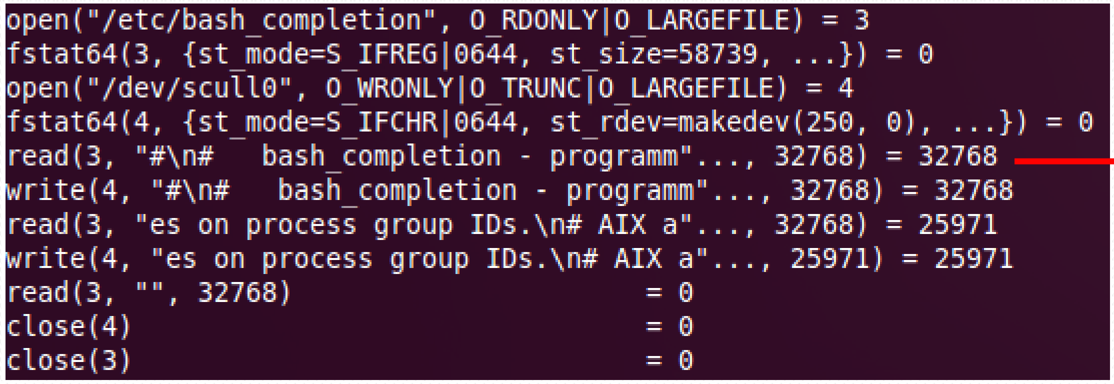
- **Writing more than one quantum (size of the file is 58739 bytes):**
  - *cp /etc/bash_completion /dev/scull0*
- **Tracing the system calls:**
  - *strace cp /etc/bash_completion /dev/scull0*


default block size for reading/writing
using 4000 byte sized quantums for writing

```
open("/etc/bash_completion", O_RDONLY|O_LARGEFILE) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=58739, ...}) = 0
open("/dev/scull0", O_WRONLY|O_TRUNC|O_LARGEFILE) = 4
fstat64(4, {st_mode=S_IFCHR|0644, st_rdev=makedev(250, 0), ...}) = 0
read(3, "#\n#   bash_completion - programm"..., 32768) = 32768
write(4, "#\n#   bash_completion - programm"..., 32768) = 4000
write(4, "ular\ncomplete -f -X '!*.@(?(e)ps"..., 28768) = 4000
write(4, "ulky functions in memory if we d"..., 24768) = 4000
write(4, "# Default to cword unchanged\n   "..., 20768) = 4000
write(4, "nt to return host:path and not o"..., 16768) = 4000
write(4, " We messed up! At least return t"..., 12768) = 4000
write(4, "ut\n# the bash < 4 compgen hack.\n"..., 8768) = 4000
write(4, "@]}' -- \"$cur\" ) )\n     __ltrim_c"..., 4768) = 4000
write(4, "o expand\n    # ~foo/... to /home"..., 768) = 768
read(3, "es on process group IDs.\n# AIX a"..., 32768) = 25971
write(4, "es on process group IDs.\n# AIX a"..., 25971) = 3232
write(4, "on completes on user or user:gro"..., 22739) = 4000
write(4, "word breaks. See __reassemble_co"..., 18739) = 4000
write(4, " /etc/ssh/ssh_config \"${HOME}/.s"..., 14739) = 4000
write(4, "    #if [[ ${COMP_KNOWN_HOSTS_WI"..., 10739) = 4000
write(4, "   # shift COMP_WORDS elements a"..., 6739) = 4000
write(4, "cal tmp\n\n    toks=( ${toks[@]-} "..., 2739) = 2739
read(3, "", 32768)                      = 0
close(4)                                = 0
close(3)                                = 0
```

Using scull

- **<u>Writing more than the capacity of the device:</u>**
  - *strace cp /usr/bin/inkscape /dev/scull0*
- **<u>Testing with quantum size 32768:</u>**
  - *rmmod scull*
  - *insmod ./scull.ko scull_quantum=32768*
  - *strace cp /etc/bash_completion /dev/scull0*



```
open("/etc/bash_completion", O_RDONLY|O_LARGEFILE) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=58739, ...}) = 0
open("/dev/scull0", O_WRONLY|O_TRUNC|O_LARGEFILE) = 4
fstat64(4, {st_mode=S_IFCHR|0644, st_rdev=makedev(250, 0), ...}) = 0
read(3, "#\n#   bash_completion - programm"..., 32768) = 32768
write(4, "#\n#   bash_completion - programm"..., 32768) = 32768
read(3, "es on process group IDs.\n# AIX a"..., 32768) = 25971
write(4, "es on process group IDs.\n# AIX a"..., 25971) = 25971
read(3, "", 32768)                       = 0
close(4)                                 = 0
close(3)                                 = 0
```

Each block is written in a single write process when quantum size is the same with the block size.

References

- Corbet, J., Rubini, A., & Kroah-Hartman, G. (2005). Chapter 3: Char Drivers. In *Linux Device Drivers, Third Edition* (pp. 42-72). O'Reilly.