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
- Three 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,
};
```

and signature of scull_ioctl is updated

```
long scull_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
{

    int err = 0, tmp;
    int retval = 0;
```

```
 /* 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);
```

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

Compiling scull

<asm/system.h> is not available, use *<asm/switch_to.h>* instead

- **One change is necessary** for Linux kernel versions newer than 3.3.

```
#include <asm/switch_to.h>      /* cli(), *_flags */
#include <asm/uaccess.h>    /* copy_*_user */
```

- During compilation, **if you get an error message saying copy_from_user and copy_to_user is undefined** (for Linux kernel versions newer than 4.1)

Replace
#include <asm/uaccess.h>
with
#include <linux/uaccess.h>

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 /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

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

Using scull

- **Writing more than one quantum (size of the file is 261KB):**
  - *cp ../ps3.pptx /dev/scull0*
- **Tracing the system calls:**
  - *strace cp ../ps3.pptx /dev/scull0*

```
open("../ps3.pptx", O_RDONLY|O_LARGEFILE) = 3
fstat64(3, {st_mode=S_IFREG|0770, st_size=266846, ...}) = 0
open("/dev/scull0", O_WRONLY|O_TRUNC|O_LARGEFILE) = 4
fstat64(4, {st_mode=S_IFCHR|0644, st_rdev=makedev(250, 0), ...}) = 0
fadvise64_64(3, 0, 0, POSIX_FADV_SEQUENTIAL) = 0
read(3, "PK\3\4\24\0\6\0\10\0\0\0!\0004lM9\r\2\0\0\24\21\0\0\23\0\10\2[C"..., 65536) = 65536
write(4, "PK\3\4\24\0\6\0\10\0\0\0!\0004lM9\r\2\0\0\24\21\0\0\23\0\10\2[C"... 65536) = 4000
write(4, "\365\333W\245\27\33\2\275\3648\263\3547\303\354\17_a\26W*\354S\324\260\223\35\10\212&Y\37"..., 61536) = 4000
write(4, "\27v\363\315RG\223\273]\337\260x\267\302\356\214\23\371nq\315E#\217Au\237\362+7\337Y"..., 57536) = 4000
write(4, "Q3\r\340Vby\242\325\245\232h\177|\276\234h\3022\324+ \202\26(K\20W\7\225\231\177"..., 53536) = 4000
write(4, "\260\340\277\236\303\214\301\324\245\26\305\23.;\202Y\211\24\330\263\304\5u>\33?\25Z\305\255Dl"..., 49536) = 4000
write(4, ".xml.rels\204\217\301\n\3020\20D\357\202\377\20\366n\322z\20\221\246^D\360\340"..., 45536) = 4000
write(4, "@y8\373\237m\267W>\231p\371\340\305z\237\226\364\241\243\316\311KM\270\t\na\311\245\245q"..., 41536) = 4000
write(4, "HX<\305\241\372Q\327\370\7\346\v\270f\223\267a28<\331\37\f;\203xz\322\351\37\235D"..., 37536) = 4000
write(4, "|\353\316<\223\224 o\353\225\201b\30z\315\343\207\0221\16~*\367\215{\361\315\312\32S>+"..., 33536) = 4000
write(4, "`B:-v<\326\344\376\230\342^K\361\263j#\vA\23\226)\221\2Wx\f\332-\211\36Q"..., 29536) = 4000
write(4, "c\347\20l\322\347\342z\251D\226+\264\311\n\343\6\253\324Tm\30\344\1\254W\314\7\215\35\302\356"..., 25536) = 4000
write(4, "j\333\372\262tf\222\225}C<\6hJ\262M\233/\221\267\301\243\200\303\362\27\261\220\237\375\250\317"..., 21536) = 4000
write(4, "\20\0\1\227\t \234\271\354\235\360\332\26\236\24ux>\362\330\"\2043\217\235\347a\323{\227\242\366"..., 17536) = 4000
write(4, "$\247q.\2\",s\334\310\311l\272w\315\330\226\325\312\245\224)32\200@3\t\224c\266S"..., 13536) = 4000
write(4, "<Yf!P\377\212z^\247\253\236Z\250\23!\310\2\2 \320P\2M6\333\335\253\376\346\305\330"..., 9536) = 4000
write(4, "\270+\347\214\6\32E\240\5f[\v~<\231k\213Q\325\255\27w\352\351a\223\243E\237Z\212\177"..., 5536) = 4000
write(4, "\335*u\26\353\f\275\224\204\216\330\373\27yl\363\237\315+\326.J\271\23\200\216\334Y!'\10\264"..., 1536) = 1536
read(3, "\0\4@\0\4ZN\0\253\355\226+\30\335\3\1\20\0\1\20h7\0018\311\333\255_\364\16\4@"..., 65536) = 65536
```

default block size for reading/writing

using 4000 byte sized quantums for writing

Using scull

- **Writing more than the capacity of the device (15510 KB):**
  - *strace cp ../intel_manual.pdf /dev/scull0*
- **Testing with quantum size 65536:**
  - *rmmod scull*
  - *rm -r /dev/scull**
  - *insmod ./scull.ko scull_quantum=65536*
  - *mknod /dev/scull0 c 250 0 (Assuming your major number is 250, also create other nodes with mknod)*
  - *strace cp ../ps3.pptx /dev/scull0*

```
open("../ps3.pptx", O_RDONLY|O_LARGEFILE) = 3
fstat64(3, {st_mode=S_IFREG|0770, st_size=266846, ...}) = 0
open("/dev/scull0", O_WRONLY|O_TRUNC|O_LARGEFILE) = 4
fstat64(4, {st_mode=S_IFCHR|0644, st_rdev=makedev(250, 0), ...}) = 0
fadvise64_64(3, 0, 0, POSIX_FADV_SEQUENTIAL) = 0
read(3, "PK\3\4\24\0\6\0\10\0\0\0!\0004lM9\r\2\0\0\24\21\0\0\23\0\10\2[C"..., 65536) = 65536
write(4, "PK\3\4\24\0\6\0\10\0\0\0!\0004lM9\r\2\0\0\24\21\0\0\23\0\10\2[C"..., 65536) = 65536
read(3, "\0\4@\0\4ZN\0\253\355\226+\30\335\3\1\20\0\1\20h7\0018\311\333\255_\364\16\4@"..., 65536) = 65536
write(4, "\0\4@\0\4ZN\0\253\355\226+\30\335\3\1\20\0\1\20h7\0018\311\333\255_\364\16\4@"..., 65536) = 65536
read(3, "\314\207\240\202\344\316\207\346\n\317\207\344\265\206\356\276t\315\272S3\225\232\334\344\200\v\7\216\234\17\303"..., 65536) = 65536
write(4, "\314\207\240\202\344\316\207\346\n\317\207\344\265\206\356\276t\315\272S3\225\232\334\344\200\v\7\216\234\17\303"..., 65536) = 65536
read(3, "\237|\343\311=\333yuv\233w\331d\213f4\211\351\205\260:\213\362\34\351\31\334\337z2)Q"..., 65536) = 65536
write(4, "\237|\343\311=\333yuv\233w\331d\213f4\211\351\205\260:\213\362\34\351\31\334\337z2)Q"..., 65536) = 65536
read(3, "a9\230\315OoPCl\372,%\271\232\v\3244\220\336\0325\221:0d\341\204\24nG\360\311"..., 65536) = 4702
write(4, "a9\230\315OoPCl\372,%\271\232\v\3244\220\336\0325\221:0d\341\204\24nG\360\311"..., 4702) = 4702
read(3, "", 65536)                      = 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.