Exercise sheet 14 – Drivers

Goals:

- Build a driver
- Load/unload a driver

Exercise 14.1: Using Drivers

Change into the directory `OS_exercises/sheet_14_drivers`
You may need admin privileges for some of the exercises.

(a) Check if the `scull` module/driver is already loaded. There are two possible methods for this: a command or the file `/proc/devices`

Proposal for solution:

```
lsmod
cat /proc/devices
```

(b) Build the `scull` driver with the included `Makefile`.

Proposal for solution: `make`

(c) Open an additional shell and open (+follow) the kernel log `/var/log/kern.log` with `tail -f /var/log/kern.log`

(d) Load the module `scull.ko` into the system.

Proposal for solution: `sudo insmod scull.ko`

(e) Did something happen in the kernel log? Read the `major device number` which was printed there.

Proposal for solution: The system printed messages to the log. The last message is the major device number, for further exercises the **major device number 243** is used by this solution. The major device number is automatically generated by the system and may be different on your system.

(f) Check if the module is now loaded inside the system.

Proposal for solution: `lsmod`
`scull` should now be inside the list.

(g) Create the two device files `scull0` and `scull1`.

Proposal for solution:

```
sudo mknod scull0 c 243 0
sudo mknod scull1 c 243 1
```

(h) Change the owner of both device files to `dev` with

```
sudo chown dev scull0 scull1
```

(i) Look at the source code of `mycat.c`. What does this program?

Proposal for solution: `mycat` does read the file referenced by the parameter and prints its content into `stdout`

(j) Compile `mycat.c` into `mycat`

Proposal for solution: `gcc -o mycat mycat.c`

(k) Write a string to `scull0`, use `mycat` to copy the string from `scull0` to `scull1`. Then read the content of `scull1`.

```
echo "This is a test!" > scull0
./mycat scull0 > scull1
./mycat scull1
```

(l) Was anything written to the log file?

Proposal for solution: Every time the driver was accessed, it wrote something into the log.

(m) Unload the driver module! Check if it is now unloaded!

Proposal for solution:

```
sudo rmmod scull
lsmod
```

The module `scull` shouldn't be listed anymore.

(n) Remove both device files!

Proposal for solution: `rm scull0 scull1`

Exercise 14.2: Look into the module sources

To answer these questions, look inside `scull.c`

(a) Which user functions are supported by the driver?

Proposal for solution: The functions `open`, `release`, `read` and `write` are supported. These are defined inside the structure `struct file_operations fpos`

(b) Where does the system get to know about the supported functions?

Proposal for solution: The system gets to know these in the function `scull_init`, when the function `register_chrdev` is called with the parameter `fpos` (This is the structure with the supported functions).

(c) How and where is the memory for the device created?

Proposal for solution:
The memory space is created in the function `scull_init` when the function `kmalloc` is called.

(d) How and where is the memory for the device released?

Proposal for solution:
The memory space is freed in the function `scull_exit` when the function `kfree` is called.

(e) Which function is used to print messages into the logfile?

Proposal for solution: The function `printk` is used to print messages into the logfile.

(f) Why is it not possible to read the same data twice from the device?

Proposal for solution: After reading, the `count` is subtracted from the `position`.

(g) How is this type of reading called?

Proposal for solution: Consuming read

(h) How is it avoided that the module is being unloaded while it is in use?

Proposal for solution: The functions `try_module_get()` (used inside the function `scull_open`) and `module_put` (used inside the function `scull_release`) do modify the module usage count inside the kernel. If this module usage count is not zero, the module cannot be removed.

(i) In which mode is the module running? In kernel (system mode) or user (user mode) space?

Proposal for solution: The module runs inside the kernel (system mode) space.