Advanced Operating Systems
Assignment Week 3

Paul Lödige
Student ID: 37-229753

October 28, 2022

1 Code

```c
/**
 * @file module.c
 * @author Paul Lödige (ploedige@g.ecc.u-tokyo.ac.jp)
 * @studentid 37-229753
 * @brief kernel module for character device that says how many times the dev file has been read from
 * @version 0.1
 * @date 2022-10-24
 *
 * @copyright Copyright (c) 2022
 *
 */

#include <linux/cdev.h>
#include <linux/delay.h>
#include <linux/device.h>
#include <linux/fs.h>
#include <linux/init.h>
#include <linux/irq.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/poll.h>

/*  Prototypes - this would normally go in a .h file */
static int device_open(struct inode *, struct file *);
static int device_release(struct inode *, struct file *);
static ssize_t device_read(struct file *, char __user *, size_t, loff_t *);
static ssize_t device_write(struct file *, const char __user *, size_t,
                            loff_t *);

#define SUCCESS 0
#define DEVICE_NAME "chardev" /* Dev name as it appears in /proc/devices */
#define BUF_LEN 80 /* Max length of the message from the device */

/* Global variables are declared as static, so are global within the file. */

static int major; /* major number assigned to our device driver */

enum {
    CDEV_NOT_USED = 0,
    CDEV_EXCLUSIVE_OPEN = 1,
};

/* Is device open? Used to prevent multiple access to device */
static atomic_t already_open = ATOMIC_INIT(CDEV_NOT_USED);

static char msg[BUF_LEN + 1]; /* The msg the device will give when asked */

static struct class *cls;

static struct file_operations chardev_fops = {
    .read = device_read,
    .write = device_write,
    .open = device_open,
    .release = device_release,
};

```

```
56 static int __init chardev_init(void)
57 {
58     major = register_chrdev(0, DEVICE_NAME, &chardev_fops);
59 
60     if (major < 0) {
61         pr_alert("Registering char device failed with %d\n", major);
62         return major;
63     }
64 
65     pr_info("I was assigned major number %d.\n", major);
66 
67     cls = class_create(THIS_MODULE, DEVICE_NAME);
68     device_create(cls, NULL, MKDEV(major, 0), NULL, DEVICE_NAME);
69 
70     pr_info("Device created on /dev/%s\n", DEVICE_NAME);
71 
72     return SUCCESS;
73 }
74 
75 static void __exit chardev_exit(void)
76 {
77     pr_info("Goodbye");
78 
79     device_destroy(cls, MKDEV(major, 0));
80     class_destroy(cls);
81 
82     /* Unregister the device */
83     unregister_chrdev(major, DEVICE_NAME);
84 }
85 
86 /* Methods */
87 
88 /* Called when a process tries to open the device file, like
89  * "sudo cat /dev/chardev"
90  */
91 static int device_open(struct inode *inode, struct file *file)
92 {
93     static int counter = 0;
94 
95     if (atomic_cmpxchg(&already_open, CDEV_NOT_USED, CDEV_EXCLUSIVE_OPEN))
96         return -EBUSY;
97 
98     sprintf(msg, "I already told you %d times Hello world!\n", counter++);
99     try_module_get(THIS_MODULE);
100 
101     return SUCCESS;
102 }
103 
104 /* Called when a process closes the device file. */
105 static int device_release(struct inode *inode, struct file *file)
106 {
107     /* We're now ready for our next caller */
108     atomic_set(&already_open, CDEV_NOT_USED);
109 
110     /* Decrement the usage count, or else once you opened the file, you will
111      * never get rid of the module.
112      */
113     module_put(THIS_MODULE);
114 
115     return SUCCESS;
116 }
117 
```

```
118 /* Called when a process, which already opened the dev file, attempts to
119  * read from it.
120  */
121 static ssize_t device_read(struct file *filp, /* see include/linux/fs.h   */
122                            char __user *buffer, /* buffer to fill with data
    */
123                            size_t length, /* length of the buffer     */
124                            loff_t *offset)
125 {
126     /* Number of bytes actually written to the buffer */
127     int bytes_read = 0;
128     const char *msg_ptr = msg;
129 
130     if (!*(msg_ptr + *offset)) { /* we are at the end of message */
131         *offset = 0; /* reset the offset */
132         return 0; /* signify end of file */
133     }
134 
135     msg_ptr += *offset;
136 
137     /* Actually put the data into the buffer */
138     while (length && *msg_ptr) {
139         /* The buffer is in the user data segment, not the kernel
140          * segment so "*" assignment won't work.  We have to use
141          * put_user which copies data from the kernel data segment to
142          * the user data segment.
143          */
144         put_user(*(msg_ptr++), buffer++);
145         length--;
146         bytes_read++;
147     }
148 
149     *offset += bytes_read;
150 
151     /* Most read functions return the number of bytes put into the buffer.
    */
152     return bytes_read;
153 }
154 
155 /* Called when a process writes to dev file: echo "hi" > /dev/hello */
156 static ssize_t device_write(struct file *filp, const char __user *buff,
157                             size_t len, loff_t *off)
158 {
159     pr_alert("Sorry, this operation is not supported.\n");
160     return -EINVAL;
161 }
162 
163 module_init(chardev_init);
164 module_exit(chardev_exit);
165 
166 MODULE_LICENSE("GPL");
```

2 Output of `sudo cat /proc/devices`

```
Character devices:
1 mem
4 /dev/vc/0
4 tty
4 ttyS
5 /dev/tty
5 /dev/console
5 /dev/ptmx
7 vcs
10 misc
13 input
29 fb
116 alsa
128 ptm
136 pts
180 usb
188 ttyUSB
189 usb_device
202 cpu/msr
203 cpu/cpuid
226 drm
235 chardev
236 binder
237 hidraw
238 wwan_port
239 nvme-generic
240 nvme
241 aux
242 bsg
243 watchdog
244 remoteproc
245 ptp
246 pps
247 cec
248 lirc
249 rtc
250 dma_heap
251 dax
252 dimmctl
253 ndctl
254 gpiochip
```

```
Block devices:
8 sd
65 sd
66 sd
67 sd
68 sd
69 sd
70 sd
71 sd
128 sd
129 sd
130 sd
131 sd
132 sd
133 sd
134 sd
135 sd
259 blkext
```

3 Output of `cat /dev/chardev`

First run:
```
I already told you 0 times Hello world!
```
Second run:
```
I already told you 1 times Hello world!
```

4 Output of `sudo dmesg -T -l info | tail -3`

```
[Fri Oct 28 15:47:13 2022] I was assigned major number 235.
[Fri Oct 28 15:47:13 2022] Device created on /dev/chardev
[Fri Oct 28 15:47:13 2022] Goodbye
```