

Отчёт по лабораторной работе №8 по дисциплине GNU/Linux

Андрей Бареков

December 13, 2019

1 Задание

Драйвер поддерживает чтение и запись сообщений в него через существующие утилиты POSIX:

```
1 echo "message text_message" > /dev/chardev
2
```

Хранит только одно сообщение. Поддерживает функцию удаления сообщения: *msg_delete*.

2 Цель работы

Драйвер, соответствующий требованиям, указанным в задании.

3 Задачи

1. Написать код драйвера.
2. Собрать драйвер.
3. Подключить драйвер.
4. Проверить правильность работы драйвера.

4 Платформы

4.1 Аппаратная платформа

CPU: Intel i5-8250U (8) @ 3.400GHz

4.2 Программная платформа

OS: Linux Manjaro x86_64

5 Ход работы

5.1 Реализация драйвера. Изменения.

Для удобства код драйвера был разделён на модули:

- **chardev.h** - основной модуль, в котором содержится документация драйвера
- **initialize.c** - модуль с инициализацией и деинициализацией драйвера
- **open.c** - модуль с открытием драйвера
- **release.c** - модуль с закрытием драйвера
- **read.c** - модуль с чтением
- **write.c** - модуль с записью

Изменено не очень приятное оформление исходного кода.
direction теперь для большей ясности является **enum**.
#ifdef DEBUG внутри функций был изменен на макрос.
Изменён алгоритм чтения и записи в соответствии с заданием.
Код модулей приведён в приложении.

5.2 Сборка и подключение модуля

Создан Makefile:

```
1 obj-m += chardev.o
2 chardev-objs := initialize.o open.o release.o read.o write.o
```

Сборка модуля и подключение:

```
1 sudo make -C /lib/modules/$(uname -r)/build M=$PWD modules
2 sudo insmod chardev.ko
3
4 ##DMESG##
5 [45189.952399] SUCCESS: Registered the major device 235
6 [45189.952407] Please, create a device file to interact with
   the driver. Use:
7 [45189.952408] mknod chardev c 235 0
8
```

5.3 Проверка корректности работы

Создан файл устройства для взаимодействия с драйвером:

```
1 sudo mknod chardev c 235 0
2
```

Проверка:

```
1 su
2 cd /dev
3 echo "message message" > chardev
4 cat chardev
5 ##RES## : message
6 echo "direction back" > chardev
7 cat chardev
8 ##RES## : egassem
9 echo "direction back" > chardev
10 cat chardev
11 ##RES## : message
12 echo "msg_delete" > chardev
13 cat chardev
14 ##RES## :
15
16 ##DMESG##
17 [45190.112732] Message has been written
18 [45190.112733] Read 5 bytes, 101207 left
19 [45190.112734] Forwards direction
20 [45190.112735] Read 5 bytes, 101207 left
21 [45190.112736] Backwards direction
22 [45190.112737] Read 5 bytes, 101207 left
23 [45190.112738] Message has been deleted
24
```

6 Выводы

В ходе работы был реализован и подключен драйвер. Взаимодействие с драйвером таким образом эффективно, поскольку можно взаимодействовать с ним, не изменяя код.

7 Приложение

7.1 Исходный код модуля chardev.h

```
1 #ifndef CHARDEV_H
2 #define CHARDEV_H
3
4 #define SUCCESS 0
5 #define BUFFER_LENGTH 256
6
7 #define DEVICE_NAME "chardev"
8
9 #include <linux/kernel.h>
10 #include <linux/module.h>
11 #include <linux/fs.h>
12 #include <linux/uaccess.h>
13
14 int init_module(void);
15 void cleanup_module(void);
16
17 static int device_open(struct inode*, struct file*);
18 static int device_release(struct inode*, struct file*);
19 static ssize_t device_read(struct file*, char*, size_t, loff_t
    *);
20 static ssize_t device_write(struct file*, const char*, size_t,
    loff_t*);
21
22 static int deviceOpen = 0;
23 static char message[BUFFER_LENGTH];
24 static char* messagePtr;
25 static int MAJOR;
26
27 enum DIRECTION
28 {
29     FORWARDS,
30     BACKWARDS
31 };
32
33 static int dir = 0;
34
35 static struct file_operations fops =
36 {
37     .read = device_read,
38     .write = device_write,
39     .open = device_open,
40     .release = device_release
41 };
42
43 MODULE_LICENSE("GPL");
44
45 #endif
```

7.2 Исходный код модуля initalize.c

```
1 #include "chardev.h"
2
3 int init_module()
4 {
5     MAJOR = register_chrdev(0, DEVICE_NAME, &fops);
6
7     if (MAJOR < 0)
8     {
9         printk(KERN_ALERT "ERROR: %s failed with %d\n", "
10             Registering the character device ", MAJOR);
11
12         return MAJOR;
13     }
14
15     printk(KERN_INFO "SUCCESS: %s%d\n", "Registered the major
16         device ", MAJOR);
17     printk(KERN_INFO "Please, create a device file to interact
18         with the driver. Use:\n");
19     printk(KERN_INFO "mknod %s c %d 0\n", DEVICE_NAME, MAJOR);
20
21     return SUCCESS;
22 }
23
24 void cleanup_module()
25 {
26     unregister_chrdev(MAJOR, DEVICE_NAME);
27     printk(KERN_INFO "Unregistered the device %d\n", MAJOR);
28 }
```

7.3 Исходный код модуля open.c

```
1 #include "chardev.h"
2
3 #ifdef DEBUG
4     #define DEBUG_INFO(file) printk(KERN_INFO "device_open(%p)\n"
5         , file)
6 #else
7     #define DEBUG_INFO
8 #endif
9
10 static int device_open(struct inode* inode, struct file* file)
11 {
12     DEBUG_INFO(file);
13
14     if (deviceOpen)
15     {
16         return -EBUSY;
17     }
18
19     ++deviceOpen;
```

```

19
20     messagePtr = message;
21     try_module_get(THIS_MODULE);
22
23     return SUCCESS;
24 }

```

7.4 Исходный код модуля release.c

```

1 #include "chardev.h"
2
3 #ifdef DEBUG
4     #define DEBUG_INFO(inode, file) printk(KERN_INFO "
5         device_release(%p, %p)\n", inode, file)
6 #else
7     #define DEBUG_INFO
8 #endif
9
10 static int device_release(struct inode* inode, struct file*
11     file)
12 {
13     DEBUG_INFO(inode, file);
14
15     --deviceOpen;
16
17     module_put(THIS_MODULE);
18
19     return SUCCESS;
20 }

```

7.5 Исходный код модуля read.c

```

1 #include "chardev.h"
2
3 #ifdef DEBUG
4     #define DEBUG_INFO(file, buffer, length) printk(KERN_INFO "
5         device_read(%p, %p, %d)\n", inode, file)
6 #else
7     #define DEBUG_INFO
8 #endif
9
10 static ssize_t device_read(struct file* file, char __user*
11     buffer, size_t length, loff_t* offset)
12 {
13     int bytes_read = 0;
14
15     if (*messagePtr == 0)
16     {
17         return 0;
18     }
19 }

```

```

17
18 if (dir == FORWARDS)
19 {
20     while (length && *messagePtr)
21     {
22         put_user(*(messagePtr++), buffer++);
23         --length;
24         ++bytes_read;
25     }
26 }
27 else
28 {
29     char* tmpPtr = messagePtr - 1;
30     while (*messagePtr)
31     {
32         ++messagePtr;
33     }
34     --messagePtr;
35
36     while (length && (messagePtr != tmpPtr))
37     {
38         put_user(*(messagePtr--), buffer++);
39         --length;
40         ++bytes_read;
41     }
42
43     *messagePtr = '\0';
44 }
45
46 printk(KERN_INFO, "Read %d bytes, %lu left\n", bytes_read,
47         length);
48 return bytes_read;
49 }

```

7.6 Исходный код модуля write.c

```

1 #include "chardev.h"
2
3 const char* DirStr = "direction";
4 const char* FrwStr = "forward";
5 const char* BckStr = "back";
6
7 const char* MsgStr = "message";
8 const char* DelStr = "msg_delete";
9
10 static ssize_t device_write(struct file* file, const char
11     __user* buffer, size_t length, loff_t* offset)
12 {
13     int i = 0;
14     char command[256];
15     copy_from_user(command, buffer, length);

```



```

15  command[length] = '\0';
16
17  if (strncmp(command, DirStr, strlen(DirStr)) == 0)
18  {
19      i = strlen(DirStr);
20
21      while ((i < length) && (command[i] == ' '))
22      {
23          ++i;
24      }
25
26      if (strncmp(command + i, FrwStr, strlen(FrwStr)) == 0)
27      {
28          dir = FORWARDS;
29          printk(KERN_INFO "Forwards direction\n");
30      }
31      else if (strncmp(command + i, BckStr, strlen(BckStr)) == 0)
32      {
33          dir = BACKWARDS;
34          printk(KERN_INFO "Backwards direction\n");
35      }
36      else
37      {
38          printk(KERN_INFO "Direction undefined\n");
39      }
40  }
41  else if (strncmp(command, MsgStr, strlen(MsgStr)) == 0)
42  {
43      i = strlen(MsgStr);
44
45      while ((i < length) && (command[i] == ' '))
46      {
47          ++i;
48      }
49
50      strcpy(message, command + i);
51      printk(KERN_INFO "Message has been written\n");
52  }
53  else if (strncmp(command, DelStr, strlen(DelStr)) == 0)
54  {
55      message[0] = '\0';
56      printk(KERN_INFO "Message has been deleted\n");
57  }
58  else
59  {
60      printk(KERN_INFO "Command undefined\n");
61  }
62
63  messagePtr = message;
64
65  return length;
66 }

```