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

Реализовать и подключить 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

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

Код модулей приведён в приложении.

5.2 Сборка и подключение модуля

Создан **Makefile**:

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

Сборка модуля и подключение:

```
sudo make -C /lib/modules/‘uname -r‘/build M=$PWD modules
sudo insmod chardev.ko

##DMESG##
[45189.952399] SUCCESS: Registered the major device 235
[45189.952407] Please, create a device file to interact with
    the driver. Use:
[45189.952408] mknod chardev c 235 0

```

5.3 Проверка корректности работы

Создан файл устройства для взаимодействия с драйвером:

```
sudo mknod chardev c 235 0

```

Проверка:

```
su
cd /dev
echo "message message" > chardev
cat chardev
##RES## : message
echo "direction back" > chardev
cat chardev
##RES## : egassem
echo "direction back" > chardev
cat chardev
##RES## : message
echo "msg_delete" > chardev
cat chardev
##RES## :

##DMESG##
[45190.112732] Message has been written
[45190.112733] Read 5 bytes, 101207 left
[45190.112734] Forwards direction
[45190.112735] Read 5 bytes, 101207 left
[45190.112736] Backwards direction
[45190.112737] Read 5 bytes, 101207 left
[45190.112738] Message has been deleted

```

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

```
#ifndef CHARDEV_H
#define CHARDEV_H

#define SUCCESS 0
#define BUFFER_LENGTH 256

#define DEVICE_NAME "chardev"

#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <linux/uaccess.h>

int init_module(void);
void cleanup_module(void);

static int device_open(struct inode*, struct file*);
static int device_release(struct inode*, struct file*);
static ssize_t device_read(struct file*, char*, size_t, loff_t
    *);
static ssize_t device_write(struct file*, const char*, size_t,
    loff_t*);

static int deviceOpen = 0;
static char message[BUFFER_LENGTH];
static char* messagePtr;
static int MAJOR;

enum DIRECTION
{
  FORWARDS,
  BACKWARDS
};

static int dir = 0;

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

MODULE_LICENSE("GPL");

#endif
```

7.2 Исходный код модуля initalize.c

```
#include "chardev.h"

int init_module()
{
  MAJOR = register_chrdev(0, DEVICE_NAME, &fops);

  if (MAJOR < 0)
  {
    printk(KERN_ALERT "ERROR: %s failed with %d\n", "
    Registering the character device ", MAJOR);

    return MAJOR;
  }

  printk(KERN_INFO "SUCCESS: %s%d\n", "Registered the major
    device ", MAJOR);
  printk(KERN_INFO "Please, create a device file to interact
    with the driver. Use:\n");
  printk(KERN_INFO "mknod %s c %d 0\n", DEVICE_NAME, MAJOR);

  return SUCCESS;
}

void cleanup_module()
{
  unregister_chrdev(MAJOR, DEVICE_NAME);
  printk(KERN_INFO "Unregistered the device %d\n", MAJOR);
}
```

7.3 Исходный код модуля open.c

```
#include "chardev.h"

#ifdef DEBUG
  #define DEBUG_INFO(file) printk(KERN_INFO "device_open(%p)\n"
    , file)
#else
  #define DEBUG_INFO
#endif

static int device_open(struct inode* inode, struct file* file)
{
  DEBUG_INFO(file);

  if (deviceOpen)
  {
    return -EBUSY;
  }

  ++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 "
       device_release(%p, %p)\n", inode, file)
5  #else
6    #define DEBUG_INFO
7  #endif
8
9  static int device_release(struct inode* inode, struct file*
       file)
10 {
11   DEBUG_INFO(inode, file);
12
13   --deviceOpen;
14
15   module_put(THIS_MODULE);
16
17   return SUCCESS;
18 }
```

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 "
       device_read(%p, %p, %d)\n", inode, file)
5  #else
6    #define DEBUG_INFO
7  #endif
8
9  static ssize_t device_read(struct file* file, char __user*
       buffer, size_t length, loff_t* offset)
10 {
11   int bytes_read = 0;
12
13   if (*messagePtr == 0)
14   {
15     return 0;
16   }
```

```
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,
        length);
47
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
       __user* buffer, size_t length, loff_t* offset)
11 {
12   int i = 0;
13   char command[256];
14   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  }
```