Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет о лабораторной работе

Курс: Проектирование ОС и компонентов

Тема: Драйвер символьного устройства

Выполнил студент группы 13541/3 _____ Д.В. Круминьш
(подпись)

Преподаватель _____ Е.В. Душутина
(подпись)

Санкт-Петербург
2018 г.

Содержание

1 Цель работы

Анализ структуры драйвера, его места в операционной системе. Написание простейшего драйвера для символьного устройства, встраивание его в операционную систему и анализ его функционирования.

2 План работы

1. Получение объектного файла драйвера - *.ko;
2. Загрузка объектного модуля *.ko в ядро;
3. Создание символьного устройства;
4. Изменение доступа к файлу символьного устройства;
5. Запись в файл устройства;
6. Чтение файла устройства;
7. Выгрузка модуля из ядра ОС;
8. Удаление файла символьного устройства;
9. Приведение каталога, в котором производилась сборка драйвера к исходному виду.

3 Сведения о системе

Работа производилась на виртуальной системе - **Ubuntu 10.04**.
Версия компилятора **gcc** - **4.4.3**.
Версия ядра - **2.6.32-21**.

4 Выполнение работы

4.1 Анализ

Драйвер устройства – это программное обеспечение, с помощью которого осуществляется управление устройством. Драйвер учитывает как специфические особенности аппаратуры, которую он контролирует, так и особенности операционной системы, в которой он функционирует.

Можно сказать, что драйвер состоит из двух частей:

1. является специфической для конкретного устройства;
2. является специфической для ОС.

Часть драйвера устройства, характерная для конкретного устройства, будет одной и той же во всех операционных системах и в большей мере она связана с анализом и пониманием спецификаций устройства, а не с программированием.

Также, в зависимости от контролируемого устройства, драйверы бывают нескольких типов:

- **Символьные драйверы** – программное обеспечение, которое управляет доступом к символьному устройству (клавиатура, мышь и т.п), которое генерирует или воспринимает поток символов (байты);
- **Блочные драйверы** – программное обеспечение, которое управляет доступом к блочному устройству (жесткий диск), которые оперируют блоками данных;

Если посмотреть на схему с пространстрами ОС Linux, то можно заметить, что драйвер

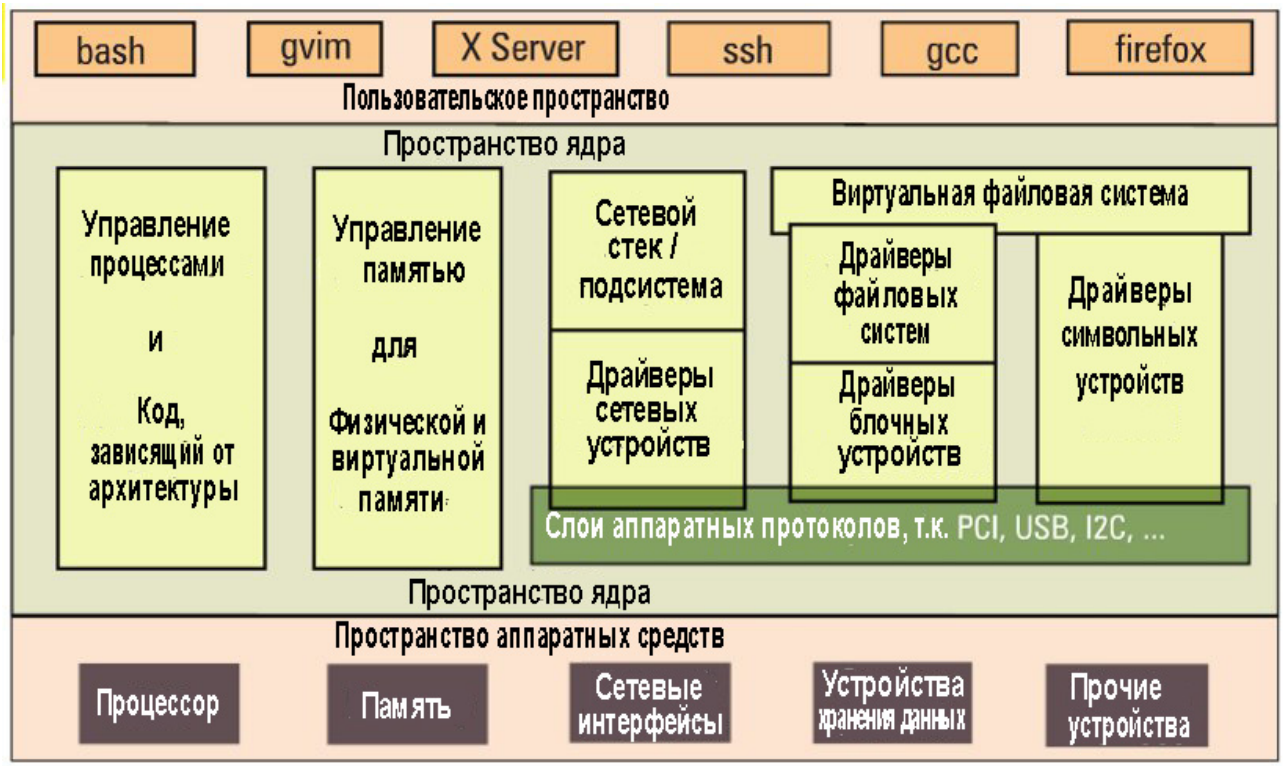

Рис. 1: Пространства ОС Linux

является частью ядра. Это сделано для возможности доступа к аппаратной части.

То есть драйвер - это объектный файл, загруженный в ядро, который предоставляет набор функций для доступа к устройству из ОС.

4.2 Реализация драйвера

4.2.1 Компиляция

Исходный код драйвера символьного устройства приведен в приложении к отчету. Структура драйвера является базовой, и может быть использована для создания драйверов для более сложных устройств.

Для получения объектного модуля, производится компиляция и сборка исходного кода драйвера с помощью утилиты make (соответствующий makefile находится в приложении к отчету):

```
1 psaer@ubuntu:~/Desktop/driver_1$ make
2 make -C/lib/modules/2.6.32-21-generic/build M=/home/psaer/Desktop/driver_1
    ↪ modules
3 make[1]: Entering directory '/usr/src/linux-headers-2.6.32-21-generic'
```

```
4   CC [M]  /home/psaer/Desktop/driver_1/chardev.o
5 /home/psaer/Desktop/driver_1/chardev.c:38: warning: initialization from
    ↪ incompatible pointer type
6 /home/psaer/Desktop/driver_1/chardev.c: In function "device_read:
7 /home/psaer/Desktop/driver_1/chardev.c:135: warning: format '%d' expects type
    ↪ "int", but argument 3 has type "size_t
8 /home/psaer/Desktop/driver_1/chardev.c: In function "device_write:
9 /home/psaer/Desktop/driver_1/chardev.c:144: warning: format '%d' expects type
    ↪ "int", but argument 4 has type "size_t
10   Building modules, stage 2.
11   MODPOST 1 modules
12   CC      /home/psaer/Desktop/driver_1/chardev.mod.o
13   LD [M]  /home/psaer/Desktop/driver_1/chardev.ko
14 make[1]: Leaving directory '/usr/src/linux-headers-2.6.32-21-generic'
```

Листинг 1: Лог сборки объектного файла драйвера

По итогу компиляции, появляется файл с раширением **.ko**. Дополнительно, с помощью команды **modinfo** можно получить информуцию о модуле.

```
1 psaer@ubuntu:~/Desktop/driver_1$ modinfo chardev.ko
2 filename:       chardev.ko
3 srcversion:     A15D07213C10C205BEB39CD
4 depends:
5 vermagic:       2.6.32-21-generic SMP mod_unload modversions
```

Листинг 2: Информация о модуле

4.2.2 Встраивание в ядро

Для загрузки модуля в ядро, используется команда **insmod** с правами суперпользователя.

```
1 psaer@ubuntu:~/Desktop/driver_1$ sudo insmod chardev.ko
2 [sudo] password for psaer:
3 psaer@ubuntu:~/Desktop/driver_1$ tail /var/log/syslog
4 May  8 10:28:01 ubuntu kernel: [ 1405.821348] [0]: VMCI: Updating context from
    ↪ (ID=0xfe596972) to (ID=0xfe596972) on event (type=0).
5 May  9 02:19:29 ubuntu kernel: [ 1890.047193] [0]: VMCI: Updating context from
    ↪ (ID=0xfe596972) to (ID=0xfe596972) on event (type=0).
6 May  9 02:19:50 ubuntu kernel: [ 1911.112267] chardev: module license '
    ↪ unspecified' taints kernel.
7 May  9 02:19:50 ubuntu kernel: [ 1911.112289] Disabling lock debugging due to
    ↪ kernel taint
```

```
8  May  9 02:19:50 ubuntu kernel: [ 1911.131291] I was assigned major number 250.
       ↪ To talk to
9  May  9 02:19:50 ubuntu kernel: [ 1911.131292] the driver, create a dev file
       ↪ with
10 May  9 02:19:50 ubuntu kernel: [ 1911.131293] 'mknod /dev/chardev c 250 0'.
11 May  9 02:19:50 ubuntu kernel: [ 1911.131294] Try various minor numbers. Try to
       ↪  cat and echo to
12 May  9 02:19:50 ubuntu kernel: [ 1911.131295] the device file.
13 May  9 02:19:50 ubuntu kernel: [ 1911.131295] Remove the device file and module
       ↪  when done.
```

Листинг 3: Встраивание модуля и системный лог

Дополнительно была выведена часть системного лога **/var/log/syslog**. Из лога видно, что были присвоены старший(250) и младший(0) номера драйвера, которые необходимы для связывания файла символьного устройства с встроенным драйвером.

Дополнительно, с помощью команды **lsmod**, можно проверить загрузку модуля.

```
1 psaer@ubuntu:~/Desktop/driver_1$ lsmod |grep chardev
2 chardev                3175  0
```

Листинг 4: Проверка загрузки модуля

4.2.3 Создание символьного устройства

Создать файл символьного устройства можно с помощью команды **mknod**, указав в ней старший и младший номера драйвера.

```
1 psaer@ubuntu:~/Desktop/driver_1$ sudo mknod /dev/chardev c 250 0
2 [sudo] password for psaer:
```

Листинг 5: Создание символьного устройства

4.2.4 Изменение прав доступа

Изменение прав доступа необходимо для обеспечения возможности записи и чтения файла символьного устройства. Для этого необходимо использовать команду **chmod**.

```
1 psaer@ubuntu:~/Desktop/driver_1$ sudo chmod 666 /dev/chardev
2 psaer@ubuntu:~/Desktop/driver_1$ ls -l /dev/chardev
3 crw-rw-rw- 1 root root 250, 0 2018-05-09 02:40 /dev/chardev
```

Листинг 6: Изменение прав доступа

4.2.5 Запись в файл и чтение из файла устройства

Запись в файл символьного устройства осуществлялась с помощью команды **echo** и перенаправления потока ввода в файл символьного устройства. Чтение выполнялось с помощью команды **cat**.

```
1 psaer@ubuntu:~/Desktop/driver_1$ echo 'hello' > /dev/chardev
2 psaer@ubuntu:~/Desktop/driver_1$ cat /dev/chardev
3 I already told you 1 times HELLO!
```

Листинг 7: Записиь и чтение в файл символьного устройства

Откроем содержимое системного лога **/var/log/syslog**.

```
1  May  9 03:15:33 ubuntu kernel: [ 5251.389870] Try to open character device /dev
       ↪ /chardev c 250 0'.
2  May  9 03:15:33 ubuntu kernel: [ 5251.389884] Try to write (ffff8800174b12c0,
       ↪ hello
3  May  9 03:15:33 ubuntu kernel: [ 5251.389891] Try to close character device /
       ↪ dev/chardev c 250 0'.
4  May  9 03:15:40 ubuntu kernel: [ 5258.156384] Try to open character device /dev
       ↪ /chardev c 250 0'.
5  May  9 03:15:40 ubuntu kernel: [ 5258.156398] Try to read character device /dev
       ↪ /chardev c 250 0'.
6  May  9 03:15:40 ubuntu kernel: [ 5258.156400] Read 34 bytes, 32734 left.
7  May  9 03:15:40 ubuntu kernel: [ 5258.156405] Try to read character device /dev
       ↪ /chardev c 250 0'.
8  May  9 03:15:40 ubuntu kernel: [ 5258.156406] Try to close character device /
       ↪ dev/chardev c 250 0'.
9  May  9 03:16:05 ubuntu kernel: [ 5283.911003] Try to open character device /dev
       ↪ /chardev c 250 0'.
10 May  9 03:16:05 ubuntu kernel: [ 5283.949311] Try to write (ffff88003c70f440,
       ↪ hello
11 May  9 03:16:05 ubuntu kernel: [ 5283.949312] , 6).
12 May  9 03:16:05 ubuntu kernel: [ 5283.949400] Try to close character device /
       ↪ dev/chardev c 250 0'.
```

Листинг 8: Часть системного лога

Из данного лога, становиться понятным, что при записи совершается: **открытие, запись, закрытие** файла, а при чтении: **открытие, чтение, закрытие** файла.

4.2.6 Запись в файл и чтение из файла устройства (пользовательская программа)

Также работа с символьным устройством была произведена с помощью пользовательской программы(прикреплена в приложении), суть которой заключается в выполнении команд **open, read, write, close**.

```
psaer@ubuntu:~/Desktop/driver_1$ ./userdev.o
We work with character device driver
Character driver open
Character driver write 50 bytes
Character driver read 24 bytes
Hello, character device!
Character driver close
```

Листинг 9: Лог пользовательской программы

Откроем содержимое системного лога **/var/log/syslog**.

```
May  9 03:47:58 ubuntu kernel: [ 7195.605496] Try to open character device /dev
    ↪ /chardev c 250 0'.
May  9 03:47:58 ubuntu kernel: [ 7195.605501] Try to write (ffff88003c65c240,
    ↪ Hello, character device!, 50).
May  9 03:47:58 ubuntu kernel: [ 7195.605509] Try to read character device /dev
    ↪ /chardev c 250 0'.
May  9 03:47:58 ubuntu kernel: [ 7195.605510] Read 24 bytes, 26 left.
May  9 03:47:58 ubuntu kernel: [ 7195.605513] Try to close character device /
    ↪ dev/chardev c 250 0'.
```

Листинг 10: Часть системного лога

Модуль сработал корректно, как и ожидалось, команды **open, read, write, close** были выполнены именно в этом порядке.

4.2.7 Выгрузка модуля

Для выгрузки модуля из ядра системы используется команда **rmmod** с правами суперпользователя.

```
psaer@ubuntu:~/Desktop/driver_1$ sudo rmmod chardev.ko
[sudo] password for psaer:
psaer@ubuntu:~/Desktop/driver_1$ tail /var/log/syslog --lines=1
May  9 03:58:52 ubuntu kernel: [ 7848.985196] Removing character device /dev/
    ↪ chardev c 250 0'.
```

Листинг 11: Выгрузка модуля

В логе **/var/log/syslog** также появилась соответствующая запись.

4.2.8 Удаление файла символьного устройства

Для удаления файла символьного устройства используется команда **rm**.

```
1 psaer@ubuntu:~/Desktop/driver_1$ sudo rm /dev/chardev
2 psaer@ubuntu:~/Desktop/driver_1$ ls /dev | grep chardev
```

Листинг 12: Удаление файла символьного устройства

4.2.9 Приведение каталога сборки к исходному виду

Для очистки каталога от исполняемых файлов используется сценарий **clean** из **Makefile**.

```
1  psaer@ubuntu:~/Desktop/driver_1$ ls
2  chardev.c  chardev.mod.c  chardev.o  modules.order  userdev.c
3  chardev.ko  chardev.mod.o  Makefile  Module.symvers  userdev.o
4  psaer@ubuntu:~/Desktop/driver_1$ make clean
5  make -C/lib/modules/2.6.32-21-generic/build M=/home/psaer/Desktop/driver_1
       ↪ clean
6  make[1]: Entering directory '/usr/src/linux-headers-2.6.32-21-generic'
7    CLEAN   /home/psaer/Desktop/driver_1/.tmp_versions
8    CLEAN   /home/psaer/Desktop/driver_1/Module.symvers /home/psaer/Desktop/
       ↪ driver_1/modules.order
9  make[1]: Leaving directory '/usr/src/linux-headers-2.6.32-21-generic'
10 psaer@ubuntu:~/Desktop/driver_1$ ls
11 chardev.c  Makefile  userdev.c
```

Листинг 13: Удаление файла символьного устройства

Вывод

Итогом данной работы является:

1. Изучение базовой структуры драйвера для ОС Linux;
2. Анализ функционирования драйвера символьного устройства на примере простейшего драйвера;
3. Получение базовых знаний и навыков встраивания/выгрузки модулей в ядро операционной системы.

Список литературы

[1] Драйверы устройств. — URL: http://www.interface.ru/home.asp?artId=29352 (дата обращения: 2018-05-08).

[2] Написание драйверов в Linux (linux driver gcc). — URL: https://www.opennet.ru/base/dev/linux_driver.txt.html (дата обращения: 2018-05-08).

Приложение 1

Исходный код драйвера символьного устройства chardev.c

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

#include <linux/slab.h>
#include <linux/errno.h>
#include <linux/types.h>

#define SUCCESS 0
#define DEVICE_NAME "chardev"   /* Имя устройства, будет отображаться в /proc/
    ↪ devices   */
#define BUF_LEN 80      /* Максимальная длина сообщения */

/* Прототипы функций */
int device_init(void);
void device_exit(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 *, char *, size_t, loff_t *);

module_init(device_init);
module_exit(device_exit);

/* Глобальные переменные – объявлены как статические для избегания конфликтов */

static int Major;       /* Старший номер устройства – драйвера */
static int Device_Open = 0;     /* Счетчик открытия устройства
                 * Используется для предотвращения обращения
                 * из нескольких процессов */

static char Message[BUF_LEN];   /* Текст сообщения */
static char *Message_Ptr;

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

```
41 };
42
43 /* Загрузка модуля в ядро */
44 int device_init(void)
45 {
46     /* Регистрация устройства */
47         Major = register_chrdev(0, DEVICE_NAME, &fops);
48
49     if (Major < 0) {
50       printk(KERN_ALERT "Registering char device failed with %d\n", Major);
51       return Major;
52     }
53
54     printk(KERN_INFO "I was assigned major number %d. To talk to\n", Major);
55     printk(KERN_INFO "the driver, create a dev file with\n");
56     printk(KERN_INFO "'mknod /dev/%s c %d 0'.\n", DEVICE_NAME, Major);
57     printk(KERN_INFO "Try various minor numbers. Try to cat and echo to\n");
58     printk(KERN_INFO "the device file.\n");
59     printk(KERN_INFO "Remove the device file and module when done.\n");
60
61     return SUCCESS;
62 }
63
64 /* Выгрузка модуля из ядра */
65 void device_exit(void)
66 {
67     /* Освобождение старшего номера устройства */
68     unregister_chrdev(Major, DEVICE_NAME);
69     printk(KERN_ALERT "Removing character device /dev/%s c %d 0'.\n",
    ↪      DEVICE_NAME, Major);
70     return;
71 }
72
73 /* Методы */
74
75 /* Открытие файла устройства процессом "cat /dev/chardev" */
76 static int device_open(struct inode *inode, struct file *file)
77 {
78     static int counter = 0;
79     printk(KERN_INFO "Try to open character device /dev/%s c %d 0'.\n",
    ↪          DEVICE_NAME, Major);
80     if (Device_Open)
81       return -EBUSY;
82
83     Device_Open++; // счетчик открытия устройства
```

```
84      sprintf(Message, "I already told you %d times HELLO!\n", counter++);
85      Message_Ptr = Message;
86      try_module_get(THIS_MODULE);
87
88      return SUCCESS;
89  }
90
91  /* Закрытие файла устройства процессом */
92  static int device_release(struct inode *inode, struct file *file)
93  {
94      printk(KERN_ALERT "Try to close character device /dev/%s c %d 0'.\n",
    ↪           DEVICE_NAME, Major);
95      Device_Open--;  /* Возможно обслуживание другого процесса */
96
97      /*
98       * Уменьшить счетчик обращений, иначе после успешного
99       * открытия не сможем больше выгрузить модуль.
100      */
101     module_put(THIS_MODULE);
102     return SUCCESS;
103 }
104
105 /* Открытие файла устройства процессом для чтения */
106 static ssize_t device_read(struct file *filp, /*указатель на структуру file*/
107         char *buffer,   /* буфер, куда надо положить данные */
108         size_t length,  /* длина буфера */
109         loff_t * offset)
110 {
111     /* Количество байт, записанных в буфер */
112     int bytes_read = 0;
113     printk("Try to read character device /dev/%s c %d 0'.\n", DEVICE_NAME,
    ↪ Major);
114
115     /*
116      * Если достигли конца сообщения,
117      * вернуть ноль, как признак конца файла
118      */
119     if (*Message_Ptr == 0)
120         return 0;
121
122     /* Помещение данных в буфер*/
123     while (length && *Message_Ptr) {
124           /*
125      * Буфер находится в пространстве пользователя в( сегменте данных),
126      * а не в пространстве ядра, поэтому простое присваивание
```

```
127         * здесь недопустимоДля. того чтобы скопировать данные, используем        *
        ↪ функцию put_user, которая перенесет данные из пространства ядра в
128         * пространство пользователя.
129         */
130             put_user(*(Message_Ptr++), buffer++);
131             length--;
132             bytes_read++;
133         }
134
135         printk("Read %d bytes, %d left.\n", bytes_read, length);
136         /* Возвращаем число байт, записанных в буфер */
137         return bytes_read;
138 }
139
140 /* Открытие файла устройства процессом для записи */
141 static ssize_t device_write(struct file *filp, char *buffer, size_t length,
        ↪ loff_t * offset)
142 {
143         int i;
144         printk("Try to write (%p, %s, %d).\n", filp, buffer, length);
145         for (i = 0; i < length && i < BUF_LEN; i++)
146             get_user(Message[i], buffer + i);
147         Message_Ptr = Message;
148         return i;
149 }
```

Листинг 14: chardev.c

Приложение 2

Makefile

```
1 obj-m += chardev.o
2
3 all:
4         make -C/lib/modules/$(shell uname -r)/build M=$(PWD) modules
5 clean:
6         make -C/lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Листинг 15: Makefile

Приложение 3

Пользовательская программа для работы с символьным устройством

```c
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <stdlib.h>
#include <stdio.h>
#include <linux/unistd.h>

int main()
{
    int fd;
    size_t cnt = 0;
    size_t cnt2 = 0;
    char bufferIn[50] = "Hello, character device!";
    char bufferOut[50];

    printf("We work with character device driver\n");
    fd = open("/dev/chardev", O_RDWR);
    if (fd == -1) {
        printf("open failed\n");
        return -1;
    }
    printf("Character driver open\n");

    cnt = write(fd, bufferIn, sizeof(bufferIn));
    printf("Character driver write %d bytes\n", cnt);

    cnt = read(fd, bufferOut, sizeof(bufferOut));
    printf("Character driver read %d bytes\n", cnt);
    int i =0;
    while ( i<cnt){
        printf("%c",bufferOut[i]);
        i++;
    }
    close(fd);
    printf("\nCharacter driver close\n");
    return 0;
}
```

Листинг 16: userdev.c