**实验五 驱动程序开发**

**一、      实验目的**

1.     掌握Linux环境下设备分类及设备文件的查看；

2.     理解主设备号和次设备号的概念；

3.     掌握设备驱动程序设计流程；

4.     掌握驱动程序相关的数据结构

5.     掌握驱动程序设计的主要工作；

6.     掌握Linux驱动程序的加载、卸载、查看方法。

**二、      实验工具和环境**

PC机、Linux Ubuntu操作系统

**三、      实验内容与结果**

编写字符设备驱动程序my\_driver.c，并编写测试程序（test.c）通过读写该设备调用驱动程序，完成对字符设备内容的读取和写入。其中my\_driver.c文件中要求完成字符设备的打开、关闭等内核态和用户态之间的接口，同时定义函数，实现字符内容的逆序写入。在test.c中完成字符数组的定义，并把该数组中的内容写入设备文件，以及从设备文件中读取部分内容，验证驱动程序的正确性。

**四、实验结果**

将以上综合案例的代码复制到下面，并将执行的输出结果截图放到代码后面。

My\_driver

#include <linux/init.h>

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/slab.h>

#include <linux/fs.h>

#include <linux/errno.h>

#include <linux/types.h>

#include <linux/proc\_fs.h>

#include <linux/fcntl.h>

#include <linux/poll.h>

#include <asm/uaccess.h>

static int charMajor = 0;

static char drv\_buffer[100];

static int char\_open(struct inode \*inode, struct file \*filp) {

printk("Device open success!\n");

return 0;

}

static int char\_release(struct inode \*inode, struct file \*filp) {

printk("Device release successed\n");

return 0;

}

static ssize\_t char\_write(struct file \*filp, const char \_\_user \*buffer, size\_t count, loff\_t \*offset) {

if (count > 100)

count = 100;

if (copy\_from\_user(drv\_buffer, buffer, count)) {

return -EFAULT;

}

int i, j;

char temp;

for (i = 0, j = count - 1; i < j; i++, j--) {

temp = drv\_buffer[i];

drv\_buffer[i] = drv\_buffer[j];

drv\_buffer[j] = temp;

}

return count;

}

static ssize\_t char\_read(struct file \*filp, char \_\_user \*buffer, size\_t count, loff\_t \*offset) {

if (count > 100)

count = 100;

if (copy\_to\_user(buffer, drv\_buffer, count)) {

return -EFAULT;

}

return count;

}

static struct file\_operations char\_fops = {

owner: THIS\_MODULE,

write: char\_write,

read: char\_read,

open: char\_open,

release:char\_release,

};

static int \_\_init char\_init(void) {

int ret = register\_chrdev(0, "char", &char\_fops);

if (ret < 0) {

printk("can't get major number\n");

return ret;

}

charMajor = ret;

return 0;

}

static void \_\_exit char\_exit(void) {

unregister\_chrdev(charMajor, "char");

printk("Exit device successfully!\n");

return;

}

module\_init(char\_init);

module\_exit(char\_exit);

MODULE\_LICENSE("GPL");

MODULE\_AUTHOR("ZHANG");

Test.c:

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <unistd.h>

#define DEVICE\_FILE\_NAME "/dev/char"

int main() {

int fd;

char write\_buf[100] = "Hello, this is a test string.";

char read\_buf[100];

fd = open(DEVICE\_FILE\_NAME, O\_RDWR);

if (fd < 0) {

perror("Failed to open the device...");

return errno;

}

printf("Writing to the device: %s\n", write\_buf);

if (write(fd, write\_buf, sizeof(write\_buf)) < 0) {

perror("Failed to write to the device...");

close(fd);

return errno;

}

if (read(fd, read\_buf, sizeof(read\_buf)) < 0) {

perror("Failed to read from the device...");

close(fd);

return errno;

}

printf("Read from the device: %s\n", read\_buf);

close(fd);

return 0;

}









