МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №5 з дисципліни

«Системне програмування С і С++»

Варіант 2

|  |  |
| --- | --- |
| Перевірив:  Сокульський О.Є. | Виконав:  студент 3 курсу  групи ІП-11 ФІОТ  Прищепа В.С. |

Київ-2023

**Лабораторна робота №5**

**Створення віртуального символьного пристрою у варіанті модуля ядра Linux і прототипів системних викликів**

**Завдання:**

1. Встановити та налаштувати середовище для розрбки модулів ядра.

2. Створити новий проект користуючись додатком. У разі появи помилок внести зміни до вихідного коду.

3. Після успішного виконання п. 1, 2 завантажити модуль ядра.

4. Переконатися у його коректній роботі.

5. Створити програму користувача. У разі появи помилок внести зміни до вихідного коду.

6. Протестувати модуль ядра з рівня користувача.

7. Зробити висновки щодо особливостей виконання лабораторної роботи, наявності специфічних особливостей роботи програми.

**Код програми:**

**lab5.c:**

#include <linux/kernel.h>

#include <linux/init.h>

#include <linux/module.h>

#include <linux/kdev\_t.h>

#include <linux/fs.h>

#include <linux/cdev.h>

#include <linux/device.h>

#include <linux/time.h>

#include <linux/uaccess.h>

#include <linux/random.h>

MODULE\_AUTHOR("Pryshchepa Vladyslav");

MODULE\_DESCRIPTION("Simple char device module");

MODULE\_LICENSE("GPL");

#define BUF\_LEN 256

#define MSG\_PREF "MSG: "

#define print\_msg(msg, ...) printk(KERN\_ERR MSG\_PREF msg, ##\_\_VA\_ARGS\_\_);

dev\_t devt = 0;

static struct class \*dev\_class = NULL;

static struct cdev my\_cdev;

static struct device \*dev = NULL;

static int my\_open(struct inode \*inode, struct file \*file);

static int my\_release(struct inode \*inode, struct file \*file);

static ssize\_t my\_read(struct file \*filp, char \_\_user \*buf, size\_t len,loff\_t \* off);

static ssize\_t my\_write(struct file \*filp, const char \*buf, size\_t len, loff\_t \* off);

static long my\_ioctl(struct file \*filp, unsigned int cmd, unsigned long arg);

static struct file\_operations fops = {

.owner = THIS\_MODULE,

.read = my\_read,

.write = my\_write,

.open = my\_open,

.release = my\_release,

.unlocked\_ioctl = my\_ioctl,

};

static int my\_open(struct inode \*inode, struct file \*file)

{

print\_msg("Open Function Called...!!!\n");

return 0;

}

static int my\_release(struct inode \*inode, struct file \*file)

{

print\_msg("Release Function Called...!!!\n");

return 0;

}

static ssize\_t my\_read(struct file \*filp, char \_\_user \*buf, size\_t len, loff\_t \*off)

{

uint8\_t data[BUF\_LEN] = {0};

uint16\_t number;

uint8\_t bitset [sizeof(number) \* 8 + 1] = {0};

print\_msg("Read Function Called...!!!\n");

//my func

get\_random\_bytes(&number, sizeof(number));

for (int i = 0; i < sizeof(number) \* 8; i++){

if ((number & 1) << i == 0){

bitset[i] = '0';

}

else{

bitset[i] = '1';

}

number >>=1;

}

bitset[sizeof(number) \* 8] = '\0';

snprintf(data, BUF\_LEN, "Random set of bits: %s", bitset);

if (len > BUF\_LEN) {

len = BUF\_LEN;

}

if (copy\_to\_user(buf, data, len)) {

return -EFAULT;

}

return len;

}

static ssize\_t my\_write(struct file \*filp, const char \_\_user \*buf, size\_t len, loff\_t \*off)

{

print\_msg("Write Function Called...!!!\n");

return len;

}

static long my\_ioctl(struct file \*filp, unsigned int cmd, unsigned long arg) {

print\_msg("Ioctl Function Called...!!!\n");

return 0;

}

static int \_\_init cdev\_module\_init(void)

{

long res = 0;

if((res = alloc\_chrdev\_region(&devt, 0, 1, "my\_cdev")) < 0){

print\_msg("Cannot allocate major number\n");

goto alloc\_err;

}

cdev\_init(&my\_cdev, &fops);

if((res = cdev\_add(&my\_cdev, devt, 1)) < 0){

print\_msg("Cannot add the device to the system\n");

goto cdev\_add\_err;

}

dev\_class = class\_create(THIS\_MODULE, "my\_class");

if(IS\_ERR(dev\_class)){

res = PTR\_ERR(dev\_class);

print\_msg("Cannot create the struct class\n");

goto class\_err;

}

dev = device\_create(dev\_class, NULL, devt, NULL, "my\_cdev");

if(IS\_ERR(dev)){

res = PTR\_ERR(dev);

print\_msg("Cannot create the Device\n");

goto dev\_create\_err;

}

print\_msg("Device Insert...Done!!!\n");

return 0;

dev\_create\_err:

class\_destroy(dev\_class);

class\_err:

cdev\_add\_err:

unregister\_chrdev\_region(devt, 1);

alloc\_err:

return res;

}

static void \_\_exit cdev\_module\_exit(void)

{

device\_destroy(dev\_class, devt);

class\_destroy(dev\_class);

cdev\_del(&my\_cdev);

unregister\_chrdev\_region(devt, 1);

print\_msg("Device Remove...Done!!!\n");

}

module\_init(cdev\_module\_init);

module\_exit(cdev\_module\_exit);

**main.c:**

#include<stdio.h>

#include<fcntl.h>

#include<errno.h>

#include<unistd.h>

#include<sys/ioctl.h>

#define BUF\_LEN 256

int main(void) {

int fd = open("/dev/my\_cdev", O\_RDONLY);

if (fd < 0) {

printf("Error number %d", errno);

}

unsigned char buf[BUF\_LEN] = {0};

read(fd, buf, BUF\_LEN);

printf("%s\n", buf);

close(fd);

}

**Makefile:**

ifneq ($(KERNELRELEASE),)

obj-m := lab5.o

else

KDIR ?= /lib/modules/uname -r/build

default:

$(MAKE) -C $(KDIR) M=$$PWD modules

gcc -o main main.c

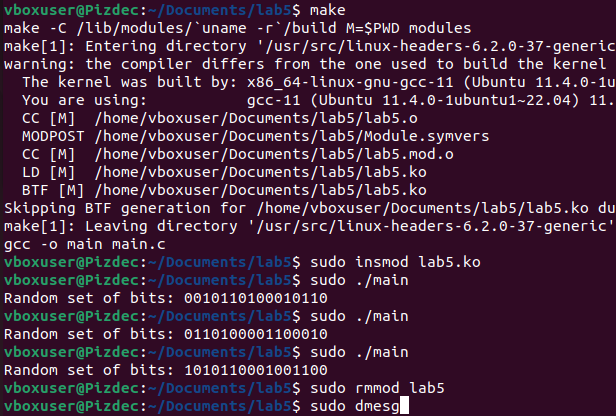
clean:

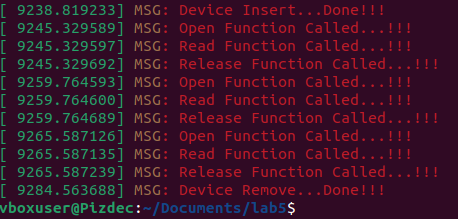
$(MAKE) -C $(KDIR) M=$$PWD clean

rm -f main

endif

**Виконання:**





**Висновок:** під час виконання лабораторної роботи я створив модуль ядра і програму, що використовує прототипи системних викликів. Код і результати виконання наведені вище.