МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВОЙ ПРОЕКТ
ЗАЩИЩЕН С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ

ст.преп.		Поляк М.Д.
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

СОЗДАНИЕ USB ДРАЙВЕРА

по дисциплине: ОПЕРАЦИОННЫЕ СИСТЕМЫ И СЕТИ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4336

	подпись, дата	инициалы, фамилия

Санкт-Петербург
2016

1 Цель работы

Цель работы: Знакомство с устройством ядра ОС Linux. Получение опыта разработки драйвера устройства.

2 Задание(4 вариант)

Добавление защиты от несанкционированного запуска операционной системы. Необходимо внести изменения в процесс загрузки ядра Linux, добавив проверку наличия подключенного через интерфейс USB flash-накопителя с заданным серийным номером. Если в процессе загрузки операционной системы нужный flash-накопитель подключен к одному из портов USB, то операционная система успешно загружается в штатном режиме. Если flash-накопитель с нужным серийным номером отсутствует, необходимо приостановить загрузку операционной системы и предоставить пользователю три попытки подключить нужный flash-накопитель. Если пользователь три раза подключает flash- накопитель с неправильным серийным номером, произвести выключение компьютера. Если же пользователь подключит нужный flash-накопитель, продолжить загрузку операционной системы в штатном режиме.

3 Техническая документация

1. Сборка проекта:
   Скачиваем файлы с репозитория на github при помощи команды:

   ```
   git clone https://github.com/NatashaMamtseva/driver.git
   ```

2. Добавление в автозагрузку:
   Шаг 1: Собираем драйвер (test.ko) с помощью запуска команды make.
   Шаг 2: Копируем файл в папку /lib/modules/версия ядра/test.ko.
   Шаг 3: Добавляем драйвер в автозагрузку с помощью команды depmod test.ko.
   Шаг 4:Отключаем флеш-устройство при загрузке системы.
   Шаг 5: Перезгружаем систему.
   Шаг 6: Все работает, система требует флешку для запуска.

4 Скриншоты

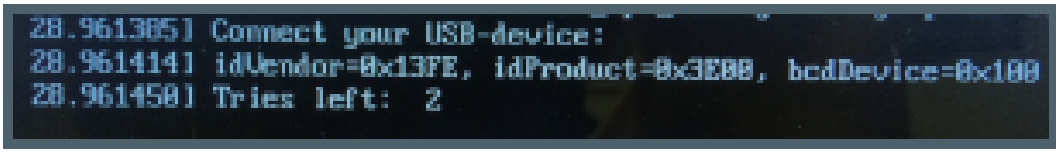

Рис. 1: Вставка неправильной флешки

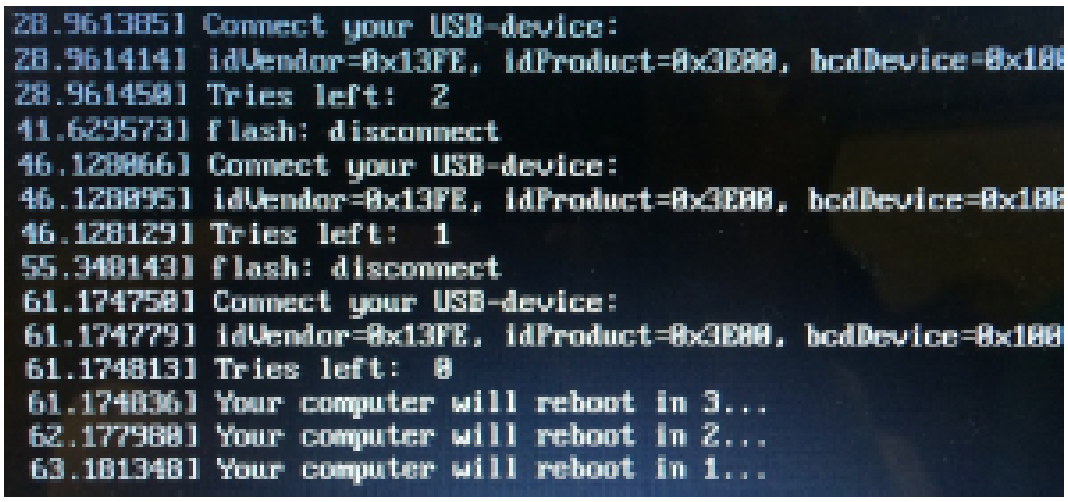

Рис. 2: Вставка неправильной флешки третий раз перезагрузка компьютера

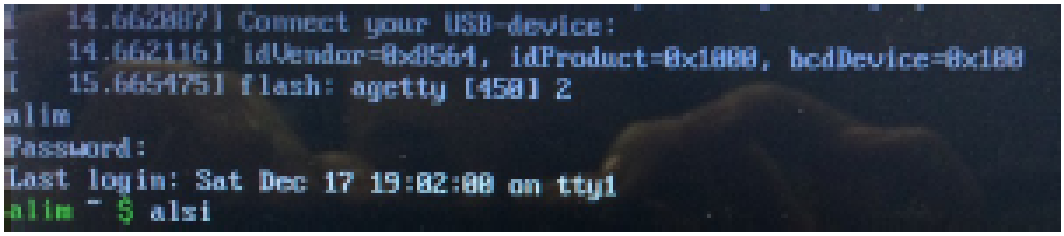

Рис. 3: Вставка коректной флешки

5 Выводы

В процессе выполнения данной курсовой работы мною были получены знания и навыки, необходимые для работы с ядром ОС Linux, а так же знания и навыки в разработке драйверов устройств.

6 Приложение

test.c:

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/usb.h>
#include <linux/sched.h>
#include <linux/kthread.h>
#include <linux/types.h>
#include <linux/tty.h>
#include <linux/version.h>
#include <linux/delay.h>
#include <linux/reboot.h>

struct task_struct *tAgetty;
struct task_struct *task;
bool stopThread = true;
bool isTry = true;
static int param = 1;
module_param( param, int, 0 );
int countTry = 3;

static int thread_agetty_uninterrupyible( void * data)
{
// основной цикл потока
while(stopThread)
{
for_each_process(task)
{

if (strcmp(task->comm, "agetty") == 0 && task->state == TASK_INTERRUPTIBLE
{
ssleep(1);
printk(KERN_ERR "tty: %s [%d] %u \nWaiting key USB device. %d attempts\n",
task->state = TASK_UNINTERRUPTIBLE;
}
}
}
if (countTry <= 0)
{
printk(KERN_ERR "Reboot in 3...\n");
ssleep(1);
printk(KERN_ERR "Reboot in 2...\n");
ssleep(1);
```

```
printk(KERN_ERR "Reboot in 1...\n");
ssleep(1);
kernel_restart(NULL);
}

return -1;
}

static int pen_probe(struct usb_interface *interface, const struct usb_dev
{
struct usb_device *dev = interface_to_usbdev(interface);

printk( KERN_ERR "USB Connected: idVendor=0x%hX, idProduct=0x%hX, Serial=%
dev->descriptor.idVendor,
dev->descriptor.idProduct, dev->serial );

if (isTry)
{
if (dev->descriptor.idVendor == 0x0951 && dev->descriptor.idProduct == 0x1
{
stopThread = false;
isTry = false;
ssleep(1);
printk( KERN_ERR "Key USB device connected\n");

for_each_process(task)
{
if (strcmp(task->comm, "agetty") == 0 && task->state == TASK_UNINTERRUPTIB
{
//printk(KERN_ERR "flash: %s [%d] %u \n", task->comm , task->pid, (u32)tas
task->state = TASK_INTERRUPTIBLE;
}
}
} else {

countTry--;
printk(KERN_ERR "Tries left:  %i \n", countTry);
if (countTry <= 0)
{
stopThread = false;
}
}
}
```

```
return 0;
}

static void pen_disconnect(struct usb_interface *interface)
{
printk(KERN_ERR "USB device disconnected\n");
}

static struct usb_device_id pen_table[] =
{
{ USB_DEVICE(0x0951, 0x1603) },
     {}
};

static struct usb_driver pen_driver =
{
.name = "usb_auth",
.probe = pen_probe,
.disconnect = pen_disconnect,
.id_table = pen_table,
};

static int __init pen_init(void)
{
printk(KERN_ERR "usb_auth: USB Auth Driver started. 3 attempts\n");

// поток блокирования tty
tAgetty = kthread_create( thread_agetty_uninterrupyible, NULL, "agetty_uni

if (!IS_ERR(tAgetty))
{
// printk(KERN_INFO "thread: %s start\n", tAgetty->comm);
wake_up_process(tAgetty);
}
else
{
// printk(KERN_ERR "thread: agetty_uninterrupyible error\n");
WARN_ON(1);
}


return usb_register(&pen_driver);
}
```

```
static void __exit pen_exit(void)
{
usb_deregister(&pen_driver);
}

module_init(pen_init);
module_exit(pen_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("none");
MODULE_DESCRIPTION("USB Auth Driver");
```