МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

**Факультет программной инженерии и компьютерной техники**

**Лабораторная работа №2**

по дисциплине

“Системы Ввода-Вывода”

Основы написания драйверов устройств с использованием операционной системы

Вариант 4

Основы написания драйверов устройств с  
использованием операционной системы

**Студент:**

Румский А.

Группа 1.1

**Преподаватель:**

Табунщик Сергей Михайлович

Санкт-Петербург, 2025 год

Задание

Написать драйвер символьного устройства, удовлетворяющий  
требованиям:  
• должен создавать символьное устройство /dev/varN, где N – это  
номер варианта  
• должен обрабатывать операции записи и чтения в соответствии с  
вариантом задания

При записи текста в файл символьного устройства должен  
осуществляться подсчет введенных цифр. Последовательность  
полученных результатов (количество цифр) с момента  
загрузки модуля ядра должна выводиться при чтении файла.

Код

Чтение

static ssize\_t dev\_read(struct file \*file, char \_\_user \*buf, size\_t count, loff\_t \*ppos) {

if (\*ppos >= results\_len) return 0;

if (count > results\_len - \*ppos) count = results\_len - \*ppos;

if (copy\_to\_user(buf, results + \*ppos, count)) return -EFAULT;

\*ppos += count;

return count;

}

Запись

static ssize\_t dev\_write(struct file \*file, const char \_\_user \*buf, size\_t count, loff\_t \*ppos) {

char \*kbuf;

int i;

int local\_digit\_count = 0;

kbuf = kmalloc(count + 1, GFP\_KERNEL); // память

if (!kbuf) return -ENOMEM;

if (copy\_from\_user(kbuf, buf, count)) { // данные в ядро

kfree(kbuf);

return -EFAULT;

}

kbuf[count] = '\0';

for (i = 0; i < count; i++) { // считаем цифры

if (kbuf[i] >= '0' && kbuf[i] <= '9') {

local\_digit\_count++;

}

}

digit\_count += local\_digit\_count;

results\_len += snprintf(results + results\_len, sizeof(results) - results\_len, "%d\n", local\_digit\_count); // сохранение

kfree(kbuf);

return count;

}

Загрузка

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

if (alloc\_chrdev\_region(&dev\_number, 0, 1, DEVICE\_NAME) < 0) {

return -1;

}

cdev\_init(&cdev, &fops);

if (cdev\_add(&cdev, dev\_number, 1) < 0) {

unregister\_chrdev\_region(dev\_number, 1);

return -1;

}

cls = class\_create(THIS\_MODULE, CLASS\_NAME);

if (IS\_ERR(cls)) {

cdev\_del(&cdev);

unregister\_chrdev\_region(dev\_number, 1);

return PTR\_ERR(cls);

}

device\_create(cls, NULL, dev\_number, NULL, DEVICE\_NAME);

printk(KERN\_INFO "var4 device driver loaded\n");

return 0;

}

Выгрузка

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

device\_destroy(cls, dev\_number);

class\_destroy(cls);

cdev\_del(&cdev);

unregister\_chrdev\_region(dev\_number, 1);

printk(KERN\_INFO "var4 device driver unloaded\n");

}

Работа

