**Лабораторная работа 2**

**По дисциплине “Системы ввода-вывода”**

“Основы написания драйверов устройств с использованием операционной системы”

Вариант 4

Выполнил:

Баянов Равиль Динарович  
Кузнецов Даниил Александрович

Поток 1.4

Санкт-Петербург

2025 г.

**Оглавление**

[**Цель** 3](#_Toc192858743)

[**Задачи** 4](#_Toc192858744)

[**Вариант** 5](#_Toc192858745)

[**Описание функций** 6](#_Toc192858746)

[**Вывод** 7](#_Toc192858747)

# **Цель**

Познакомится с основами разработки драйверов устройств с использованием операционной системы на примере создания драйверов символьных устройств под операционную систему Linux.

# **Задачи**

1. 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

2. Оформить отчет по работе в электронном формате

# **Вариант**

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.

# **Описание функций**

Функция чтения из устройства:

Возвращает количество введённых цифр при чтении из устройства.

snprintf формирует строку с количеством цифр.

copy\_to\_user копирует данные из пространства ядра в пространство пользователя.

Если данные уже были прочитаны (\*off > 0), функция завершает чтение, что реализует поведение "прочитать один раз".

|  |
| --- |
| static ssize\_t **my\_read**(**struct** file \*f, char \_\_user \*buf, size\_t len, loff\_t \*off)  {  char str[100];  ssize\_t str\_len;    str\_len = snprintf(str, sizeof(str), "Total digits entered: %d\n", digits\_counter);  **if** (len < str\_len) {  **return** -EINVAL;  }  **if** (\*off > 0) {  **return** 0;  }  **if** (copy\_to\_user(buf, str, str\_len)) {  **return** -EFAULT;  }    \*off += str\_len;  **return** str\_len;  } |

Функция записи в устройство:

Записывает данные из пространства пользователя в буфер ядра.

Считает количество цифр в записанных данных и увеличивает digits\_counter.

copy\_from\_user копирует данные из пространства пользователя в пространство ядра.

printk логирует количество принятых байтов и цифр.

|  |
| --- |
| static ssize\_t **my\_write**(**struct** file \*f, const char \_\_user \*buf, size\_t len, loff\_t \*off)  {  size\_t bytes\_to\_write = (len > BUF\_SIZE) ? BUF\_SIZE : len;  **if** (copy\_from\_user(device\_buffer, buf, bytes\_to\_write)) {  **return** -EFAULT;  }  device\_buffer[bytes\_to\_write] = '\0';  data\_len = bytes\_to\_write;  printk(KERN\_INFO "Driver: wrote %zu bytes\n", bytes\_to\_write);  char\* input\_str = device\_buffer;  int i = 0;  **while** (input\_str[i] != '\0') {  **if** (input\_str[i] >= '0' && input\_str[i] <= '9') {  digits\_counter++;  }  i++;  }  printk(KERN\_INFO "Driver: wrote %zu digits\n", digits\_counter);  device\_buffer[0] = '\0';  **return** bytes\_to\_write;  } |

# **Вывод**

Выполнив данную лабораторную работу, мы реализовали простой драйвер устройств с помощью ОС. Мы изучили как работают драйверы, работающие под операционную систему Linux.