ИТМО

Системы ввода/вывода

Лабораторная работа №2
«Основы написания драйверов устройств с использованием операционной системы»

Преподаватели
Сергей Быковский
Сергей Табунщик

2025

Лабораторная работа 2

ИТМО

- **Тема:** «Основы написания драйверов устройств с использованием операционной системы»

- **Цель:** познакомится с основами разработки драйверов устройств с использованием операционной системы на примере создания драйверов символьных устройств под операционную систему Linux.

- По для выполнения работы:
  - qemu-system-riscv32
  - ОС Linux
  - RISC-V C/ASM окружение для сборки проектов (на базе LLVM)

Материалы

ИТМО

- ✓ Классическая книга по разработке драйверов для Linux
  - J. Corbet, A. Rubini, G. Kroah-Hartman. **Linux Device Drivers**. Third Edition. 2005. P. 638
- ✓ Драйвера устройств в Linux (перевод Linux Device Drivers Series")
  - http://rus-linux.net/MyLDP/BOOKS/drivers/linux-device-drivers-00.html
- ✓ Инструменты разработки под ОС Linux для RISC-V и образ Linux для QEMU
  - https://syntacore.com/tools/development-tools
- ✓ Информация по сборке внешнего модуля ядра
  - https://www.kernel.org/doc/html/latest/kbuild/modules.html
- ✓ Информация о символьных устройствах
  - https://linux-kernel-labs.github.io/refs/heads/master/labs/device_drivers.html

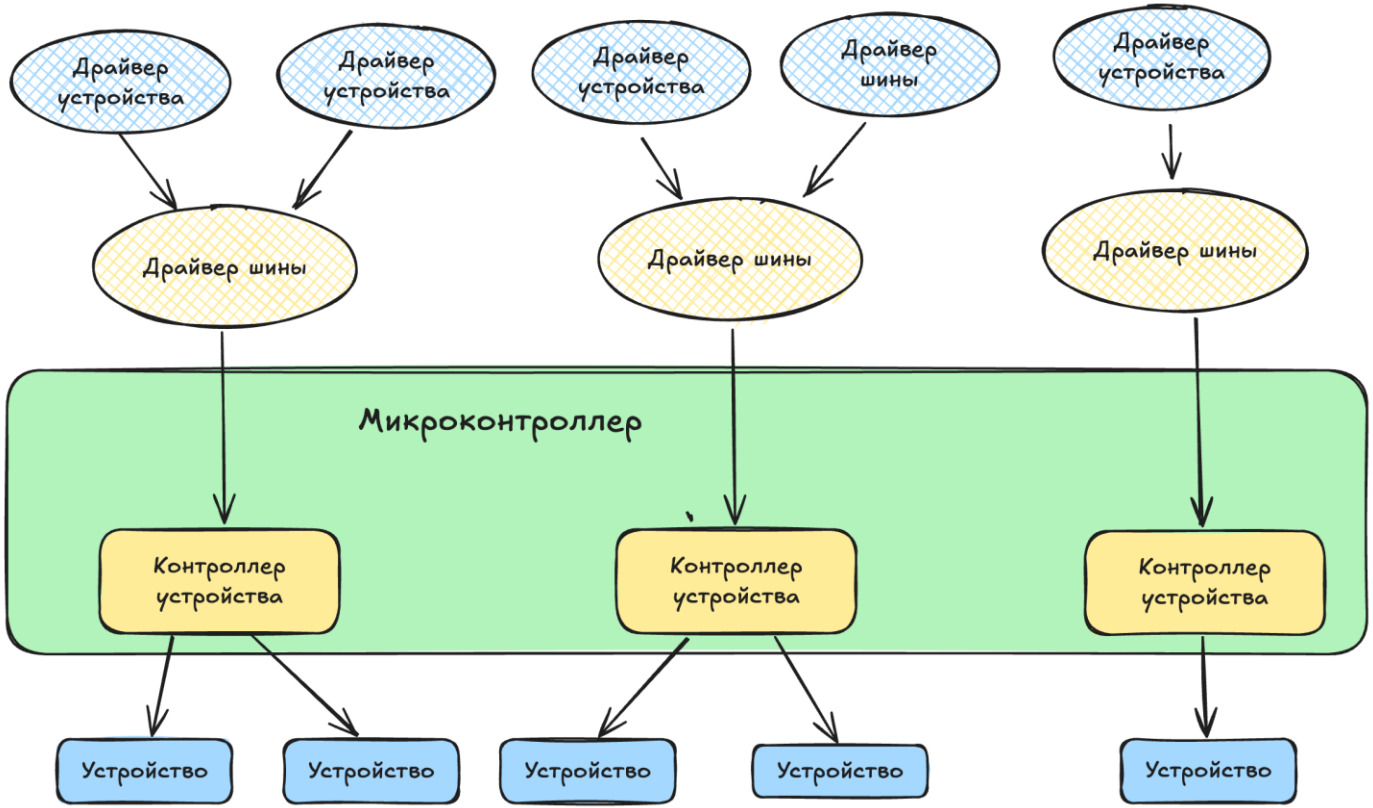
Понятие драйвера в ОС Linux

ИТМО

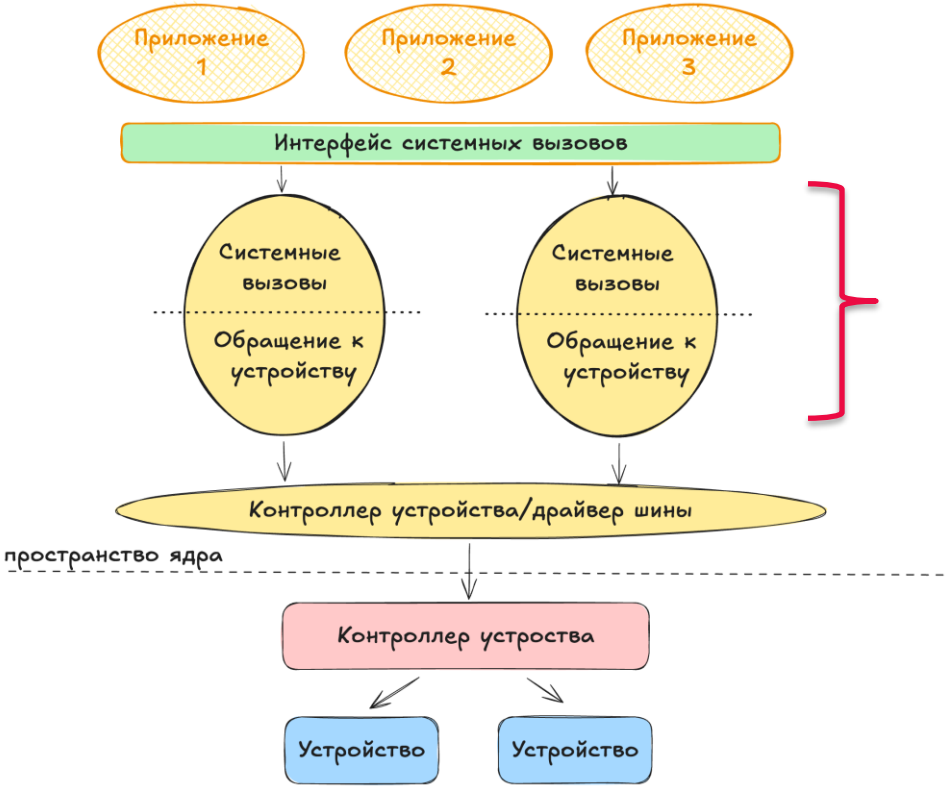
- Драйвер – это управляющая программа, предназначенная для реализации обмена данными с внешними устройствами

- Драйвера в ОС Linux реализуются в виде модулей ядра

Драйвера в ОС Linux

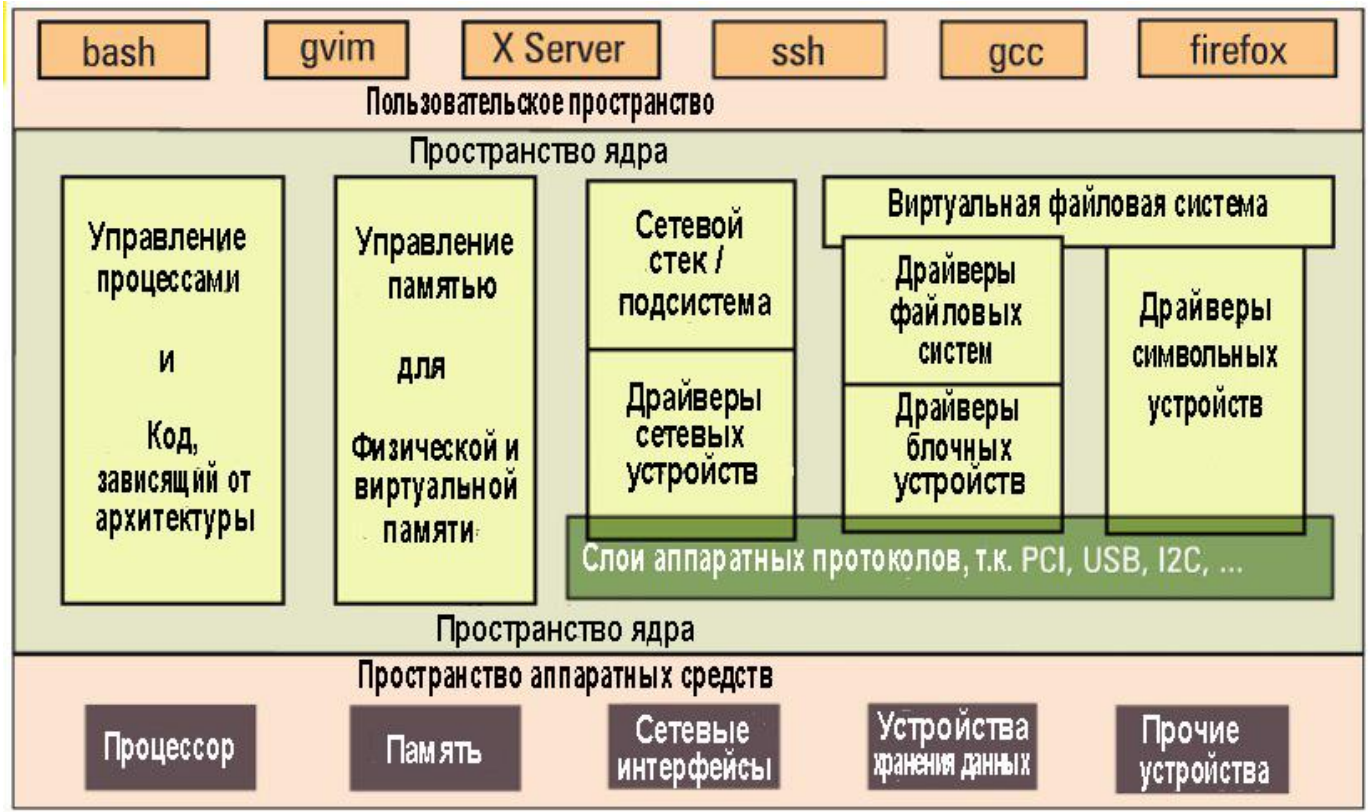


Устройство драйверов



драйвера устройств

Пространство ядра ОС Linux



Структура драйвера

ИТМО

- Заголовочные файлы ядра
- Конструктор
- Деструктор
- Код инициализации
- Информация о драйвере
- Тело драйвера

Заголовочные файлы ядра

ИТМО

```
#include <linux/module.h>
#include <linux/version.h>
#include <linux/kernel.h>
#include <linux/types.h>
#include <linux/kdev_t.h>
#include <linux/fs.h>
#include <linux/device.h>
#include <linux/cdev.h>
#include <linux/uaccess.h>
#include <linux/slab.h>
```

Конструктор

ИТМО

```
static dev_t first;

static int __init ch_drv_init(void)
{
    printk(KERN_INFO "Hello!\n");
    if (alloc_chrdev_region(&first, 0, 1, "ch_dev") < 0)
    {
        return -1;
    }

    return 0;
}
```

Деструктор

ИТМО

```
static void __exit ch_drv_exit(void)
{
    unregister_chrdev_region(first, 1);
    printk(KERN_INFO "Bye!!!\n");
}
```

Код инициализации

```
module_init(ch_drv_init);

module_exit(ch_drv_exit);
```

Информация о драйвере

ИТМО

```
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Author");
MODULE_DESCRIPTION("The first kernel module");
```

Функции обмена данными с пространством пользователя

- copy_from_user
- copy_to_user

Компиляция: содержимое Makefile

```
obj-m = ch_drv.o
PWD = $(shell pwd)
all:
	make -C /lib/modules/$(shell uname -r)/build
M="$(PWD)" modules
clean:
	make -C /lib/modules/$(shell uname -r)/build
M="$(PWD)" clean
```

Команды работы с модулем ядра

ИТМО

- **insmod** ch_drv.ko – загрузка модуля ядра
- **rmmod** ch_drv – выгрузка модуля ядра
- **modinfo** ch_drv.ko – получение информации о модуля ядра

- **lsmod** – просмотр загруженных модулей ядра.
- **dmesg** – чтение кольцевого буфера ядра.

Инструкция по установке окружения(1)

ИТМО

1. Устанавливаем opensbi и uboot для riscv64

```
sudo apt install u-boot-qemu opensbi
```

2. Скачиваем архив с образом Ubuntu20.04.5-preinstalled-server-riscv64

```
wget https://cdimage.ubuntu.com/releases/20.04/release/ubuntu-
20.04.5-preinstalled-server-riscv64+unmatched.img.xz
```

3. Распаковываем архив

```
xz -dk ubuntu-20.04.5-preinstalled-server-
riscv64+unmatched.img.xz
```

4. Расширяем обаз на 5Гб

```
qemu-img resize -f raw ubuntu-20.04.5-preinstalled-server-
riscv64+unmatched.img +5G
```

Инструкция по установке окружения(2)

ИТМО

5. Запускаем QEMU на машине virt с использованием скаченного OpenSBI в качестве BIOS, UBoot в качестве Bootloader и Ubuntu в качестве OS

```
qemu-system-riscv64 \
-machine virt \
-nographic \
-m 5G \
-smp 1 \
-bios /usr/lib/riscv64-linux-gnu/opensbi/generic/fw_jump.bin \
-kernel /usr/lib/u-boot/qemu-riscv64_smode/uboot.elf \
-device virtio-rng-pci \
-drive file=ubuntu-20.04.5-preinstalled-server-
riscv64+unmatched.img,format=raw,if=virtio \
-device virtio-net-device,netdev=net \
-netdev user,id=net,hostfwd=tcp::2222-:22
```

Настройка Ubuntu в QEMU

ИТМО

1. При первой загрузке используем эти логин и пароль:

```
login: ubuntu
pass: ubuntu
```

2. И устанавливаем новый, например:

```
new_pass: Student12345
```

3. Увеличьте ширину консоли для удобства работы:

```
stty cols 132
```

Взаимодействие с Ubuntu на QEMU

ИТМО

- Если хотите взаимодействовать с консолью вашей ОС, вместо Ubuntu на QEMU, можно подключиться к виртуальной машине с помощью ssh в **отдельном терминале**

```
ssh ubuntu@localhost -p 2222
```

- Для передачи файла в виртуальную машину QEMU используем команду

```
scp -P 2222 ch_drv.c ubuntu@localhost:/home/ubuntu/
```

- Для передачи всей папки в виртуальную машину QEMU используем команду

```
scp -P 2222 -r ch_drv ubuntu@localhost:/home/ubuntu/
```

Сборка и запуск примера

ИТМО

1. Устанавливаем пакеты для сборки модуля ядра

```
sudo apt-get update
sudo apt install build-essential
```

2. Переходим в директорию с примером и собираем

```
cd ch_drv
make
```

3. Загружаем модуль ядра

```
sudo insmod ch_drv.ko
```

3. Пишем/читаем файл устройства /dev/mychdev, читаем кольцевой буфер ядра с отладочной информацией и радуемся, что все работает ;)

Задание к лабораторной работе

ИТМО

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

Варианты (1)

ИТМО

№ варианта	Описание
1	При записи текста в файл символьного устройства должен осуществляться подсчет введенных символов. Последовательность полученных результатов (количество символов) с момента загрузки модуля ядра должна выводиться при чтении файла.
2	При записи текста в файл символьного устройства должно запоминаться количество пробелов во введенном тексте. Последовательность полученных результатов с момента загрузки модуля ядра должна выводиться при чтении файла.

Варианты (2)

ИТМО

№ варианта	Описание
3	При записи текста в файл символьного устройства должно запоминаться количество введенных букв (не буквы на считаются). Последовательность полученных результатов с момента загрузки модуля ядра должна выводиться при чтении файла.
4	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.

Требования к отчету

ИТМО

1. На титульном листе должны быть приведены следующие данные:
   a. Название дисциплины
   b. Номер и название лабораторной работы
   c. ФИО исполнителя и группа
2. Во введении указываются цели и задачи работы
3. В основной части приводится описание функций, являющихся обработчиками системных вызовов ОС Linux на чтение и запись, а также вызываемых в этих обработчиках других функций
4. Приводятся скриншоты вывода в консоль данных при всех возможных сценариях использования драйвера.

Спасибо
за внимание!

ITMO more than a UNIVERSITY