**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет информационных технологий**

**Кафедра параллельных вычислений**

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

«Низкоуровневая работа с периферийными устройствами»

Студента 2 курса, 21211 группы

**Петрова Сергея Евгеньевича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Антон Юрьевич Кудинов

Новосибирск 2022

**СОДЕРЖАНИЕ**

## ЦЕЛЬ

- *Ознакомиться с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb;*

## ЗАДАНИЕ

1. *Реализовать программу, получающую список всех подключенных к машине USB устройств с использованием libusb. Для каждого найденного устройства напечатать его класс, идентификатор производителя и идентификатор изделия.*
2. *Изучить состав и характеристики обнаруженных с помощью реализованной программ USB устройств.*
3. *Дополнить программу функцией печати серийного номера USB устройства. Для написания функции рекомендуется использовать функции libusb_open, libusb_close, libusb_get_string_descriptor_ascii для печати поля iSerialNumber дескриптора устройства.*
4. *Составить отчет по лабораторной работе. Отчет должен содержать следующие пункты:*
    - *Титульный лист;*
    - *Цель лабораторной работы;*
    - *Полный компилируемый листинг реализованной программы и команды для ее компиляции;*
    - *Описание обнаруженных USB-устройств;*
    - *Вывод по результатам лабораторной работы.*

# ОПИСАНИЕ РАБОТЫ

## Пошаговое описание выполненной работы

1. *Реализовал программу, получающую список всех подключенных к машине USB устройств с использованием libusb. Для каждого найденного устройства напечатал его класс, идентификатор производителя, идентификатор изделия и серийный номер;*
2. *Зашёл на сервер и запустил команду lsusb;*

```
evmpu@comrade:~$ lsusb
Bus 002 Device 004: ID 046d:0825 Logitech, Inc. Webcam C270
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 003: ID 1c4f:0026 SiGma Micro Keyboard
Bus 007 Device 002: ID 0458:003a KYE Systems Corp. (Mouse Systems) NetScroll+ Mini Traveler / Genius NetScroll 120
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 002: ID 0bda:0181 Realtek Semiconductor Corp. USB2.0-CRW
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

3. *Запустил программу;*

```
/home/evmpu/21211/s.petrov1/lab6/cmake-build-release/lab6
Devices found: 12

Device class        Vendor ID           Product ID          Serial number

ef                  0×046d              0×0825              95410D90
09                  0×1d6b              0×0002              0000:00:1d.7
09                  0×1d6b              0×0001              0000:00:1d.2
00                  0×1c4f              0×0026              null
00                  0×0458              0×003a              null
09                  0×1d6b              0×0001              0000:00:1d.1
09                  0×1d6b              0×0001              0000:00:1d.0
00                  0×0bda              0×0181              20060413092100000
09                  0×1d6b              0×0002              0000:00:1a.7
09                  0×1d6b              0×0001              0000:00:1a.2
09                  0×1d6b              0×0001              0000:00:1a.1
09                  0×1d6b              0×0001              0000:00:1a.0
```

## Команды для компиляции

```
evmpu@comrade:~$ /usr/bin/cmake
-DCMAKE_BUILD_TYPE=Release
-DCMAKE_C_COMPILER=/usr/bin/gcc
-DCMAKE_CXX_COMPILER=/usr/bin/g++
-G "CodeBlocks - Unix Makefiles"
-S /home/evmpu/21211/s.petrov1/lab6
-B /home/evmpu/21211/s.petrov1/lab6/cmake-build-release
```

```
evmpu@comrade:~$ /usr/bin/cmake
--build /home/evmpu/21211/s.petrov1/lab6/cmake-build-release
--target lab6
```

**Описание обнаруженных USB-устройств**

| *Device class* | |
|---|---|
| *00* | *Unclassified device* |
| *09* | *Hub* |
| *ef* | *Miscellaneous* |

| *Vendor ID* | *Product ID* | *Device* |
|---|---|---|
| *0x1d6b* | *0x0001* | *Linux Foundation, 1.1 root hub* |
| *0x1d6b* | *0x0002* | *Linux Foundation, 2.0 root hub* |
| *0x046d* | *0x0825* | *Logitech, Inc., Webcam C270* |
| *0x1c4f* | *0x0026* | *SiGma Micro, Keyboard* |
| *0x0458* | *0x003a* | *KYE Systems Corp. (Mouse Systems), NetScroll+ Mini Traveler / Genius NetScroll 120* |
| *0x0bda* | *0x0181* | *Realtek Semiconductor Corp., USB-2.0-CRW* |

## ЗАКЛЮЧЕНИЕ

*В ходе выполнения лабораторной работы:*

- *Ознакомился с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb;*

## **ПРИЛОЖЕНИЕ** (ЛИСТИНГ ПРОГРАММЫ)

*src/main.cpp*

```
#include <libusb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// print device parameters
void PrintDeviceParams(libusb_device * device);

int main()
{
    // list of device pointers
    libusb_device ** devices;

    // libusb context session
    libusb_context * context;

    // initialize the libusb library, open a session with libusb
    int res = libusb_init(&context);
    if (res < 0)
    {
        // get error message by code
        fprintf(stderr,
                "%s\n",
                libusb_error_name(res));
        return EXIT_FAILURE;
    }

    // get a list of all found USB devices
    ssize_t device_count = libusb_get_device_list(context,
                                                  &devices);
    if (device_count < 0)
    {
        fprintf(stderr,
                "%s\n",
                libusb_error_name((int)device_count));
        return EXIT_FAILURE;
    }

    printf("Devices found: %zd\n\n", device_count);
    printf("%-20s %-20s %-20s %-20s\n",
           "Device class",
           "Vendor ID",
           "Product ID",
           "Serial number\n");

    for (int i = 0; i < device_count; i++)
    {
        PrintDeviceParams(devices[i]);
    }

    // free the memory allocated when getting the list of devices
    libusb_free_device_list(devices,
                            1);
```

```
    // shut down the libusb library
    libusb_exit(context);

    return EXIT_SUCCESS;
}
void PrintDeviceParams(libusb_device * device)
{
    // device descriptor
    struct libusb_device_descriptor descriptor;

    // get device descriptor
    int status = libusb_get_device_descriptor(device,
                                              &descriptor);
    if (status != 0)
    {
        fprintf(stderr,
                "%s\n",
                libusb_error_name(status));
        return;
    }

    // device handle
    struct libusb_device_handle * handle;

    // open device and get device handle
    status = libusb_open(device,
                         &handle);
    if (status != 0)
    {
        fprintf(stderr,
                "%s\n",
                libusb_error_name(status));
        return;
    }

    unsigned char serial_number[256];
    if (handle && descriptor.iSerialNumber)
    {
        // get string descriptor of serial number fields
        // of the device descriptor
        status = libusb_get_string_descriptor_ascii
                 (handle,
                  descriptor.iSerialNumber,
                  serial_number,
                  sizeof(serial_number));
        if (status <= 0) strncpy(serial_number,
                                 "empty",
                                 sizeof(serial_number));
    }
    else strncpy(serial_number,
                 "null",
                 sizeof(serial_number));

    // close device handle
    libusb_close(handle);
```

```
    printf("%-20.2x 0x%-18.4x 0x%-18.4x %-20s\n",
           (int)descriptor.bDeviceClass,
           descriptor.idVendor,
           descriptor.idProduct,
           serial_number);
}
```

*CMakeLists.txt*

```
cmake_minimum_required(VERSION 3.16.3)

project(lab6 C)

add_executable(lab6
        src/main.c)

find_package(PkgConfig)
pkg_check_modules(libusb-1.0 REQUIRED libusb-1.0)

target_include_directories(lab6 PUBLIC
        ${libusb-1.0_INCLUDE_DIRS})
target_link_libraries(lab6 PUBLIC
        ${libusb-1.0_LIBRARIES})
```