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;*

```
→ lab6 git:(master) ✗ lsusb
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 8087:0aa7 Intel Corp. Wireless-AC 3168 Bluetooth
Bus 001 Device 005: ID 2d95:6003 vivo vivo 1907
Bus 001 Device 004: ID 0458:0186 KYE Systems Corp. (Mouse Systems) Genius DX-120 Mouse
Bus 001 Device 002: ID 04f2:b5e0 Chicony Electronics Co., Ltd VGA WebCam
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

2. *Запустил команду lsusb;*
3. *Запустил программу с правами администратора, чтобы получить доступ к USB-устройствам по чтению и записи;*

```
→ lab6 git:(master) ✗ sudo ./cmake-build-debug/lab6
Devices found: 8

Device class        Vendor ID           Product ID          Serial number

09                  0x1d6b              0x0003              0000:05:00.4
09                  0x1d6b              0x0002              0000:05:00.4
09                  0x1d6b              0x0003              0000:05:00.3
e0                  0x8087              0x0aa7              null
00                  0x2d95              0x6003              LR7XTOOJP7FAJFVS
00                  0x0458              0x0186              null
ef                  0x04f2              0xb5e0              0x0001
09                  0x1d6b              0x0002              0000:05:00.3
```

4. *Сопоставил параметры USB устройств;*

Device class	
00	*Unclassified device*
09	*Hub*
e0	*Wireless Controller*
ef	*Miscellaneous*

Vendor ID	
0x1d6b	*Linux Foundation*

0x8087	*Intel Corp.*
0x0458	*KYE Systems Corp. (Mouse Systems)*
0x04f2	*Chicony Electronics Co., Ltd*
0x2d95	*vivo*

Product ID	
0x0002	*2.0 root hub*
0x0003	*3.0 root hub*
0x0186	*Genius DX-120 Mouse*
0x0aa7	*Wireless-AC 3168 Bluetooth*
0xb5e0	*VGA WebCam*
0x6003	*vivo 1907*

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 3.22.1)

project(lab6 C)

add_executable(lab6
        src/main.c)

find_path(LIBUSB_INCLUDE_DIR
        NAMES libusb.h)
find_library(LIBUSB_LIBRARY
        NAMES usb-1.0)

target_include_directories(lab6 PUBLIC ${LIBUSB_INCLUDE_DIR})
target_link_libraries(lab6 PUBLIC ${LIBUSB_LIBRARY})
```