**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 № 6**

«низкоуровневая работа с периферийными устройствами»

студента (ки) \_\_\_\_2\_\_\_\_\_ курса, \_\_21212\_\_ группы

**Олимпиева Юрия Юрьевича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

**А. А. Артюхов**

2022 год

**Цели**

1. 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. За основу для разработки  
можно взять программу, приведенную в листинге 1.  
2. Изучить состав и характеристики обнаруженных с помощью  
реализованной программ USB устройств.  
3. Дополнить программу, реализованную в п. 2 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.

**Описание работы**

* Подключил библиотеку libusb-1.0 к проекту через Cmake файл.
* Написал программу для получения данных о подключенном USB устройстве (приложение 1).
* Составил отчёт о выполненной работе.

**Заключение**

Изучил основы низкоуровневой работы с периферийными устройствами с помощью библиотеки libusb.

**Приложение 1**

#include <libusb-1.0/libusb.h>

#include <stdio.h>

#include <stdlib.h>

#include <cstring>

void printdev(libusb\_device \*dev){

libusb\_device\_descriptor desc; // дескриптор устройства

libusb\_config\_descriptor \*config; // дескриптор конфигурации объекта

int r = libusb\_get\_device\_descriptor(dev, &desc);

if (r < 0){

fprintf(stderr,

"Ошибка: дескриптор устройства не получен, код: %d.\n",r);

return;

}

libusb\_device\_handle\*\* dev\_handle = new libusb\_device\_handle\*;

r = libusb\_open(dev, dev\_handle);

if (r < 0) {

fprintf(stderr,

"Ошибка: дескриптор устройства не может быть открыт, код: %d.\n",r);

return;

}

unsigned char data[50] = {0};

r = libusb\_get\_string\_descriptor\_ascii(\*dev\_handle, desc.iSerialNumber, data, 49);

if (r < 0) {

strcat((char\*)data, libusb\_error\_name(r));

}

// получить конфигурацию устройства

libusb\_get\_config\_descriptor(dev, 0, &config);

printf("%-16x | %-27x | %-24x | %25s\n",

(int)desc.bDeviceClass,

(int)desc.idVendor,

(int)desc.idProduct,

data

);

libusb\_close(\*dev\_handle);

libusb\_free\_config\_descriptor(config);

delete dev\_handle;

}

int main(){

libusb\_device\*\* devs; // указатель на указатель на устройство,

// используется для получения списка устройств

libusb\_context \*ctx = NULL; // контекст сессии usb

int r; // для возвращаемых значений

ssize\_t cnt; // число найденных USB-устройств

// инициализировать библиотеку usb, открыть сессию работы с usb

r = libusb\_init(&ctx);

if(r < 0) {

fprintf(stderr,

"Ошибка: инициализация не выполнена, код: %d.\n", r);

return 1;

}

// задать уровень подробности отладочных сообщений

libusb\_set\_option(ctx, LIBUSB\_OPTION\_LOG\_LEVEL);

// получить список всех найденных USB- устройств

cnt = libusb\_get\_device\_list(ctx, &devs);

if(cnt < 0) {

fprintf(stderr,

"Ошибка: список USB устройств не получен.\n");

return 1;

}

printf("найдено устройств: %ld\n", cnt);

printf("=============================="

"=============================\n");

printf("%25s", "класс устройства |");

printf("%25s", " идентификатор производителя |");

printf("%25s", " идентификатор устройства |");

printf("%25s", "серийный номер usb устройства\n");

for(ssize\_t i = 0; i < cnt; i++) { // цикл перебора всех устройств

printdev(devs[i]); // печать параметров устройства

}

printf("=============================="

"=============================\n");

// освободить память, выделенную функцией получения списка устройств

libusb\_free\_device\_list(devs, 1);

libusb\_exit(ctx); // завершить работу с библиотекой usb,

// закрыть сессию работы с usb

return 0;

}

**Makefile**

CXX:=g++

SRC:=main.cpp

WARNINGS\_FLAGS:=-Werror -Wall -pedantic -Wextra

LD\_FLAGS:=-lusb-1.0

all:

$(CXX) $(SRC) $(WARNINGS\_FLAGS) $(LD\_FLAGS)

**Build on Linux Ubuntu:**

make