###### 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, 23208 группы

**Веретенникова Андрея Александровича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Д.С. Иванишкин

Новосибирск 2024

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc18443921)

[ЗАДАНИЕ 3](#_Toc18443922)

[ОПИСАНИЕ РАБОТЫ 4](#_Toc18443923)

[ЗАКЛЮЧЕНИЕ 5](#_Toc18443924)

[Приложение 1 (2, …). *Наименование приложения* 6](#_Toc18443925)

# ЦЕЛЬ

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

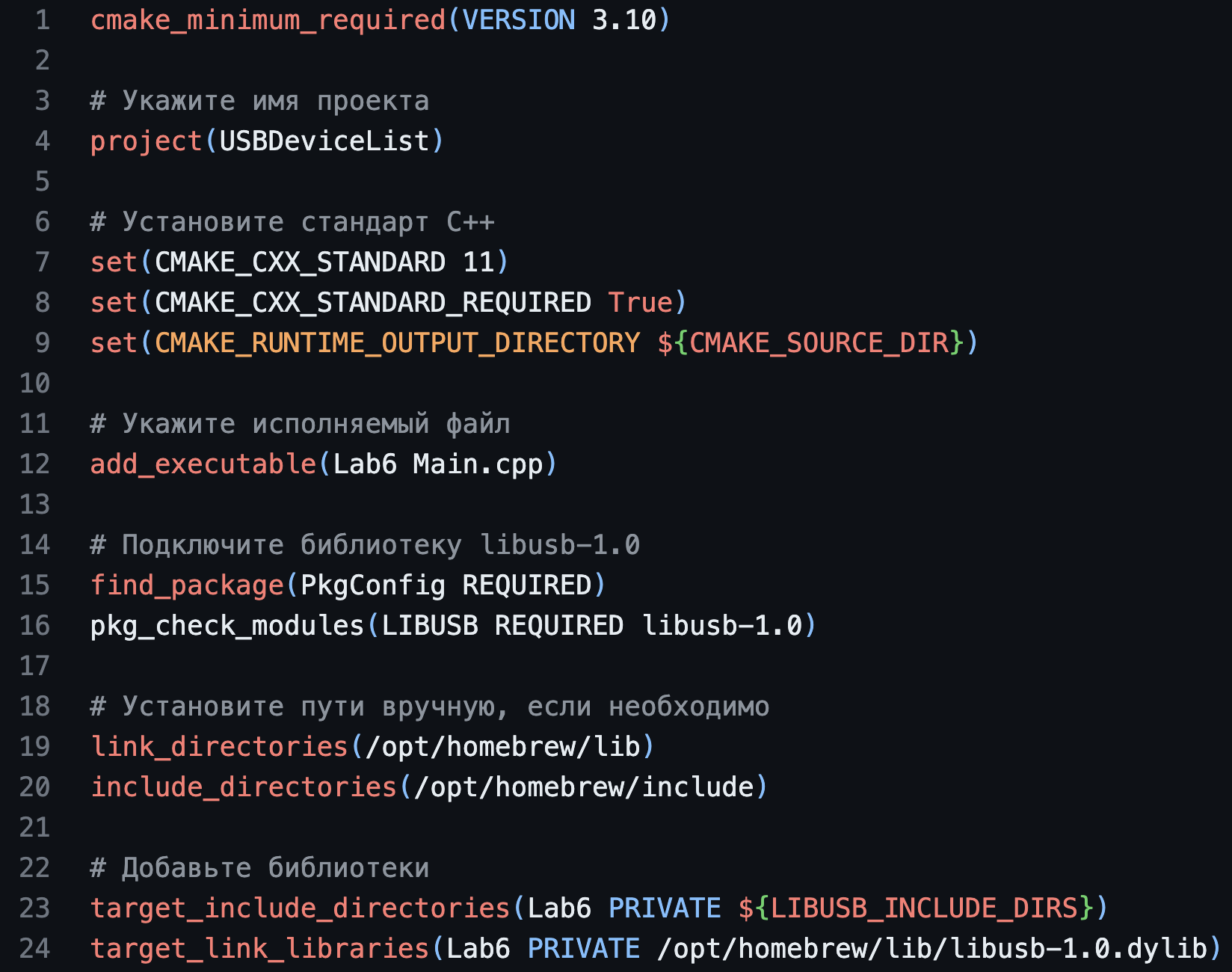
# ОПИСАНИЕ РАБОТЫ

*Пошаговое описание выполненной работы:*

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 программы USB устройств.*
3. *Дополнить программу функцией печати серийного номера USB устройства.*

*Выбранная методика: библиотека libusb.*

*Строки CMakeLists.txt для сборки проекта:*

**

*Результаты:*

# 

# ЗАКЛЮЧЕНИЕ

*В ходе лабораторной работы, я ознакомился с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb.*

# Приложение 1. *Листинг*

#include <iostream>

#include <libusb.h>

#include <stdio.h>

using namespace std;

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;

}

// получить конфигурацию устройства

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

printf("%.2d %.2d %.4d %.4d %.3d\n",

(int)desc.bNumConfigurations,

(int)desc.bDeviceClass,

desc.idVendor,

desc.idProduct,

(int)config->bNumInterfaces

);

for(int i = 0; i < (int)config->bNumInterfaces; i++) {

const libusb\_interface \*inter = &config->interface[i];

printf("| | | | | %.2d %.2d\n", inter->num\_altsetting, (int)desc.bDeviceClass);

for(int j = 0; j < inter->num\_altsetting; j++) {

const libusb\_interface\_descriptor \*interdesc = &inter->altsetting[j];

printf("| | | | | | | %.2d %.2d\n", (int)interdesc->bInterfaceNumber, (int)interdesc->bNumEndpoints);

for(int k = 0; k < (int)interdesc->bNumEndpoints; k++) {

const libusb\_endpoint\_descriptor \*epdesc = &interdesc->endpoint[k];

printf("| | | | | | | | | %.2d %.9d\n", (int)epdesc->bDescriptorType, (int)epdesc->bEndpointAddress);

}

}

}

// Открытие устройства для получения серийного номера

libusb\_device\_handle \*handle;

r = libusb\_open(dev, &handle);

if (r < 0) {

fprintf(stderr, "Ошибка: не удалось открыть устройство, код: %d.\n", r);

libusb\_free\_config\_descriptor(config);

return;

}

// Получение серийного номера

if (desc.iSerialNumber) {

unsigned char serial\_number[256];

r = libusb\_get\_string\_descriptor\_ascii(handle, desc.iSerialNumber, serial\_number, sizeof(serial\_number));

if (r < 0) {

fprintf(stderr, "Ошибка: не удалось получить серийный номер устройства, код: %d.\n", r);

} else {

printf("| | | | | | | | | | | %s\n", serial\_number);

}

}

libusb\_close(handle);

libusb\_free\_config\_descriptor(config);

}

int main() {

// используется для получения списка устройств

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

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

// инициализировать библиотеку libusb, открыть сессию работы с libusb

int 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, LIBUSB\_LOG\_LEVEL\_INFO);

// получить список всех найденных USB-устройств

// cnt - число найденных USB-устройств

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

if(cnt < 0) {

fprintf(stderr, "Ошибка: список USB устройств не получен.\n");

return 1;

}

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

printf("===========================================================\n");

printf("\* количество возможных конфигураций\n");

printf("| \* класс устройства\n");

printf("| | \* идентификатор производителя\n");

printf("| | | \* идентификатор устройства\n");

printf("| | | | \* количество интерфейсов\n");

printf("| | | | | \* количество альтернативных настроек\n");

printf("| | | | | | \* класс устройства\n");

printf("| | | | | | | \* номер интерфейса\n");

printf("| | | | | | | | \* количество конечных точек\n");

printf("| | | | | | | | | \* тип дескриптора\n");

printf("| | | | | | | | | | \* адрес конечной точки\n");

printf("| | | | | | | | | | | \* серийный номер\n");

printf("+--+--+----+----+---+--+--+--+--+--+----------------------\n");

for(int 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);

// завершить работу с библиотекой libusb

// закрыть сессию работы с libusb

libusb\_exit(ctx);

return 0;

}