###### 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

«Лабораторная работа 6»

Студента 1 курса, 19210 группы

**Пирожков Андрей Константинович**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

(ученая степень, звание)

Д.С.Иванишкин

Новосибирск 2020

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc56767257)

[ЗАДАНИЕ 3](#_Toc56767258)

[ОПИСАНИЕ РАБОТЫ 4](#_Toc56767259)

[ЗАКЛЮЧЕНИЕ 5](#_Toc56767260)

[Приложение *Листинг файла Lab6.cpp* 6](#_Toc56767261)

# 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. За основу для разработки можно взять программу, приведенную в листинге.
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. 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**

За основу этой лабораторной работы был взят код из листинга. Поэтому программа была реализована без проблем.

Использовал эту команду для компиляции (код программы в [Приложении](#_Приложение_1)):

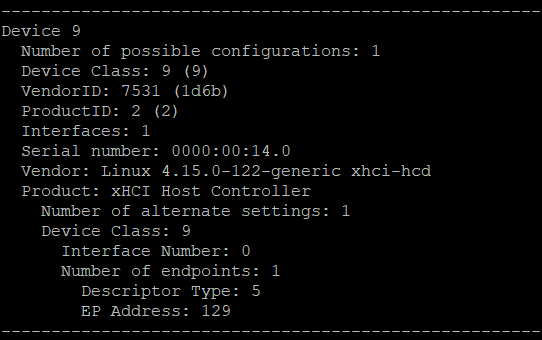
* g++ -o Lab6.exe -I/usr/include/libusb-1.0 Lab6.cpp -lusb-1.0

Я немного модифицировал код, т.к. мне не нравилось как он выводит информацию о подключенных устройствах. Также все русские символы сменил на английские, т.к. терминал не поддерживает русскую кириллицу.

Добавил вывод серийного номера, как было сказано по заданию. И вывел ещё производителя и изделие в качестве бонуса.

Не везде удалось получить данные. Возможно, им требовались права администратора. На местах, где я не смог получить данные, выводил «error» в терминал.

Прикрепляю скриншот примера вывода информации о подключенных USB устройствах:



# **ЗАКЛЮЧЕНИЕ**

В ходе этой лабораторной работы я ознакомился с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb. Я реализовал программу, получающую список всех подключённых USB устройств. Для каждого найденного устройства я печатал его класс, идентификатор производителя, идентификатор изделия, количество интерфейсов, номер интерфейса, серийный номер, название производителя, название продукта, количество альтернативных настроек, количество точек остановок и их адреса. Таким образом я изучил состав и характеристики устройств.

# Приложение

*Листинг файла Lab6.cpp*

#include <iostream>

#include <libusb.h>

#include <stdio.h>

using namespace std;

void null\_str(unsigned char\* str)

{

str[0] = 'e';

str[1] = 'r';

str[2] = 'r';

str[3] = 'o';

str[4] = 'r';

for (int i = 5; i < 64; i++)

{

str[i] = 0;

}

}

void printdev(libusb\_device\* dev)

{

libusb\_device\_descriptor desc; // дескриптор устройства

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

const libusb\_interface\* inter;

const libusb\_interface\_descriptor\* interdesc;

const libusb\_endpoint\_descriptor\* epdesc;

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

if (r < 0)

{

fprintf(stderr, "Error: device descriptor didn't received, code: %d.\n", r);

return;

}

libusb\_get\_config\_descriptor(dev, 0, &config); // получить конфигурацию устройства

printf(" Number of possible configurations: %d\n", (int)desc.bNumConfigurations); // количество возможных конфигураций

printf(" Device Class: %d (%x)\n", (int)desc.bDeviceClass, (int)desc.bDeviceClass); // класс устройства

printf(" VendorID: %d (%x)\n", desc.idVendor, desc.idVendor); // идентификатор производителя

printf(" ProductID: %d (%x)\n", desc.idProduct, desc.idProduct); // идентификатор устройства

printf(" Interfaces: %d\n", (int)config->bNumInterfaces); // количество интерфейсов

libusb\_device\_handle\* handle = NULL;

unsigned char striSerialNumber[64];

unsigned char striManufacturer[64];

unsigned char striProduct[64];

null\_str(striSerialNumber);

null\_str(striManufacturer);

null\_str(striProduct);

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

r = libusb\_claim\_interface(handle, 0);

r = libusb\_get\_string\_descriptor\_ascii(handle, desc.iSerialNumber, striSerialNumber, 64); // серийный номер

printf(" Serial number: %s\n", striSerialNumber);

r = libusb\_get\_string\_descriptor\_ascii(handle, desc.iManufacturer, striManufacturer, 64); // производитель

printf(" Vendor: %s\n", striManufacturer);

r = libusb\_get\_string\_descriptor\_ascii(handle, desc.iProduct, striProduct, 64); // изделие

printf(" Product: %s\n", striProduct);

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

{

inter = &config->interface[i];

printf(" Number of alternate settings: %d\n", inter->num\_altsetting); // количестов альтернативных настроек

printf(" Device Class: %d\n", (int)desc.bDeviceClass); // класс устройства

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

{

interdesc = &inter->altsetting[j];

printf(" Interface Number: %d\n", (int)interdesc->bInterfaceNumber); // номер интерфейса

printf(" Number of endpoints: %d\n", (int)interdesc->bNumEndpoints); // количество конечных точек

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

{

epdesc = &interdesc->endpoint[k];

printf(" Descriptor Type: %d\n", (int)epdesc->bDescriptorType); // тип дескриптора

printf(" EP Address: %d\n", (int)epdesc->bEndpointAddress); // адрес конечной точки

}

}

}

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

libusb\_free\_config\_descriptor(config);

}

int main()

{

libusb\_device\*\* devs; // указатель на указатель на устройство, используется для получения списка устройств

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

int r; // для возвращаемых значений

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

ssize\_t i; // индексная переменная цикла перебора всех устройств

r = libusb\_init(&ctx); // инициализировать библиотеку libusb, открыть сессию работы с libusb

if (r < 0)

{

fprintf(stderr, "Error: initialization failed, code: %d.\n", r);

return 1;

}

libusb\_set\_debug(ctx, 3); // задать уровень подробности отладочных сообщений

cnt = libusb\_get\_device\_list(ctx, &devs); // получить список всех найденных USB- устройств

if (cnt < 0)

{

fprintf(stderr, "Error: list of USB devices not received: %d.\n", r);

return 1;

}

printf("Devices found: %d\n", (int)cnt);

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

for (i = 0; i < cnt; i++) // цикл перебора всех устройств

{

printf("Device %d\n", (int)(i + 1));

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

}

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

libusb\_free\_device\_list(devs, 1); // освободить память, выделенную функцией получения списка устройств

libusb\_exit(ctx); // завершить работу с библиотекой libusb, закрыть сессию работы с libusb

return 0;

}