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 Бородина Артёма Максимовича 2 курса, 19205 группы

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
к.т.н, доцент
А.Ю. Власенко

Новосибирск 2020

СОДЕРЖАНИЕ

ЦЕЛЬ

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.

4. Составить отчет по лабораторной работе.

ОПИСАНИЕ РАБОТЫ

1. Изучены основы c++ библиотеки libusb версии 1.0.23 и функции данной библиотеки для работы с изображениями и видео файлами. Также были изучены стандарты usb, иерархия дескрипторов и их состав.

2. Была написана программа для получения информации о USB устройствах, подключенных к компьютеру. (Приложение 1)

3. Получаемая информация:

   - Кол-во подключенных USB устройств
   - Для каждого устройства: класс, идентификаторы производителя и изделия (их названия, если имеются), серийный номер, если имеется, кол-во конфигураций устройства
   - Для каждой конфигурации: индекс, кол-во интерфейсов устройства
   - Для каждого интерфейса: номер интерфейса, кол-во конечных точек
   - Для каждой конечной точки: тип дескриптора, адрес конечной точки

4. Для моего компьютера программа обнаружила 11 подключенных устройств (Приложение 2):
   Лишь у 4 имелся дескриптор конфигурации и все они были 0 класса, но у некоторых информация о классе имелась в дескрипторе интерфейса. Это были: мышь, клавиатура, usb-накопитель и Wi-Fi адаптер.
   Остальные 7 имели 1022 в качестве своего идентификатора производителя (Shinko Shoji Co., Ltd) и были 9 класса. Это были различного вида USB контроллеры, расположенные на материнской плате.

5. Ни у одного устройства не было серийного номера. И лишь у мыши была информация о названии производителя.

ЗАКЛЮЧЕНИЕ

После изучения библиотеки libusb были получены практические знания работы с usb устройствами и их дескрипторами.

Приложение 1. *Код программы*

```
#include <iostream>
#include "libusb.h"

using namespace std;

void printEndpoint(const libusb_endpoint_descriptor *endpoint, int num) {
    cout << "      ENDPOINT#" << num << endl;
    cout << "        bDescriptorType:   " << hex << (int) endpoint->bDescriptorType << endl;
    cout << "        bEndpointAddress:  " << (int) endpoint->bEndpointAddress<< dec << endl;
}

void printAltsetting(const libusb_interface_descriptor *interface, int num) {
    cout << "    INTERFACE#" << num << endl;
    cout << "      bInterfaceClass:     " << hex << (int) interface->bInterfaceClass << dec << endl;
    cout << "      bNumEndpoints:       " << (int) interface->bNumEndpoints << endl;
    for (int i = 0; i < interface->bNumEndpoints; i++) {
        printEndpoint(&interface->endpoint[i], i + 1);
    }
}

void printConfiguration(libusb_config_descriptor *config, int num) {
    cout << "  CONFIGURATION #" << num << endl;
    cout << "    bNumInterfaces:        " << (int) config->bNumInterfaces<< endl;
    for (int i = 0; i < (int) config->bNumInterfaces; i++) {
        const libusb_interface *inter = &config->interface[i];
        for (int t = 0; t < inter->num_altsetting; t++) {
            printAltsetting(&inter->altsetting[t], i + 1);
        }
    }

}

void printInfo(libusb_device *dev, libusb_device_descriptor desc) {
    libusb_device_handle *handle = nullptr;
    unsigned char text[128];
    libusb_open(dev, &handle);
    if (handle) {
        if (desc.iManufacturer) {
            libusb_get_string_descriptor_ascii(handle, desc.iManufacturer, text, sizeof(text));
            cout << "  Manufacturer:           " << text << endl;
        }
        if (desc.iProduct) {
            libusb_get_string_descriptor_ascii(handle, desc.iProduct, text, sizeof(text));
            cout << "  Product:                " << text << endl;
        }
        if (desc.iSerialNumber) {
            libusb_get_string_descriptor_ascii(handle, desc.iSerialNumber, text, sizeof(text));
            cout << "  Serial Number:          " << text << endl;
        } else {
            cout << "  No Serial Number" << endl;
        }
        libusb_close(handle);
    }
}
```

```
void printDevice(libusb_device *dev, int num) {
    libusb_device_descriptor desc{};

    int check = libusb_get_device_descriptor(dev, &desc);
    if (check < 0) {
        cout << "No device descriptor" << endl;
        return;
    }

    cout << endl << "Device #" << num << " (DeviceClass = " << hex
         << (int) desc.bDeviceClass << "): " << desc.idVendor << " - " << desc.idProduct
         << dec << endl;

    printInfo(dev, desc);

    for (int i = 0; i < desc.bNumConfigurations; i++) {
        libusb_config_descriptor *config;

        cout << "  bNumConfigurations:      " << (int) desc.bNumConfigurations << endl;
        check = libusb_get_config_descriptor(dev, i, &config);
        if (check != LIBUSB_SUCCESS) {
            cout << "  No config descriptor" << endl;
            continue;
        }

        printConfiguration(config, i + 1);

        libusb_free_config_descriptor(config);
    }
}

int main() {
    libusb_device **devs;

    int initCheck = libusb_init(nullptr);
    if (initCheck < 0) {
        return 1;
    }

    libusb_set_option(nullptr, LIBUSB_OPTION_LOG_LEVEL, LIBUSB_LOG_LEVEL_NONE);
    int deviceCount = libusb_get_device_list(nullptr, &devs);
    if (deviceCount < 0) {
        libusb_exit(nullptr);
        return 2;
    }
    cout << "Device count: " << deviceCount << endl;
    for (int i = 0; devs[i]; i++) {
        printDevice(devs[i], i + 1);
    }

    libusb_free_device_list(devs, 1);
    libusb_exit(nullptr);
    return 0;
}
```

Приложение 2. *Пример вывода*

```
Device count: 11
Device #1 (DeviceClass = 9): 1022 - 7807
  bNumConfigurations:   1
  No config descriptor

Device #2 (DeviceClass = 9): 1022 - 7808
  bNumConfigurations:   1
  No config descriptor

Device #3 (DeviceClass = 9): 1022 - 7807
  bNumConfigurations:   1
  No config descriptor

Device #4 (DeviceClass = 9): 1022 - 7814
  bNumConfigurations:   1
  No config descriptor

// USB-накопитель
Device #5 (DeviceClass = 0): ffff - 5678
  bNumConfigurations:   1
  CONFIGURATION #1
    bNumInterfaces:     1
    INTERFACE #1
      bInterfaceClass:  8
      bNumEndpoints:    2
      ENDPOINT #1
        bDescriptorType:    5
        bEndpointAddress:   1
      ENDPOINT #2
        bDescriptorType:    5
        bEndpointAddress:   82

Device #6 (DeviceClass = 9): 1022 - 7808
  bNumConfigurations:   1
  No config descriptor

Device #7 (DeviceClass = 9): 1022 - 7814
  bNumConfigurations:   1
  No config descriptor

// Клавиатура
Device #8 (DeviceClass = 0): 46d - c24b
  bNumConfigurations:   1
  CONFIGURATION #1
    bNumInterfaces:     2
    INTERFACE #1
      bInterfaceClass:  3
      bNumEndpoints:    1
      ENDPOINT #1
        bDescriptorType:    5
        bEndpointAddress:   81
    INTERFACE #2
      bInterfaceClass:  3
      bNumEndpoints:    1
      ENDPOINT #1
        bDescriptorType:    5
        bEndpointAddress:   82

Device #9 (DeviceClass = 9): 1022 - 7809
  bNumConfigurations:   1
  No config descriptor

// Мышь
Device #10 (DeviceClass = 0): 9da - 9090
  Manufacturer:         A4TECH
  Product:              USB Device
  No Serial Number
  bNumConfigurations:   1
  CONFIGURATION #1
    bNumInterfaces:     2
    INTERFACE #1
      bInterfaceClass:  3
      bNumEndpoints:    1
      ENDPOINT #1
        bDescriptorType:    5
        bEndpointAddress:   81
    INTERFACE #2
      bInterfaceClass:  3
      bNumEndpoints:    1
      ENDPOINT #1
        bDescriptorType:    5
        bEndpointAddress:   82

// Wi-Fi адаптер
Device #11 (DeviceClass = 0): 148f - 5370
  bNumConfigurations:   1
  CONFIGURATION #1
    bNumInterfaces:     1
    INTERFACE #1
      bInterfaceClass:  ff
      bNumEndpoints:    7
      ENDPOINT #1
        bDescriptorType:    5
        bEndpointAddress:   81
      ENDPOINT #2
        bDescriptorType:    5
        bEndpointAddress:   1
      ENDPOINT #3
        bDescriptorType:    5
        bEndpointAddress:   2
      ENDPOINT #4
        bDescriptorType:    5
        bEndpointAddress:   3
      ENDPOINT #5
        bDescriptorType:    5
        bEndpointAddress:   4
      ENDPOINT #6
        bDescriptorType:    5
        bEndpointAddress:   5
      ENDPOINT #7
        bDescriptorType:    5
        bEndpointAddress:   6
```