МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФГБОУ ВО «АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Отчет по лабораторной работе № 2

**Работа с USB-устройствами с помощью библиотеки libusb.**

Выполнил: студент 595 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Лаптев

Проверил: доц. каф. ВТиЭ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю. Я. Матющенко

Лабораторная работа защищена

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барнаул 2022

**Цель работы:** Ознакомление с библиотекой Libusb, ознакомление с интерфейсом USB, получение навыков программирования периферийных USB-устройств с использованием библиотеки LibUsb.

**Задачи:** Реализовать программу, получающую список всех подключенных к машине USB-устройств, для каждого найденного устройства напечатать все его дескрипторы. Добавить в программу возможность вывода информации об активных USB-устройствах и её динамическое обновление при подключении и отключении USB-устройств.

**Теоретические сведения:**

**Библиотека libusb**

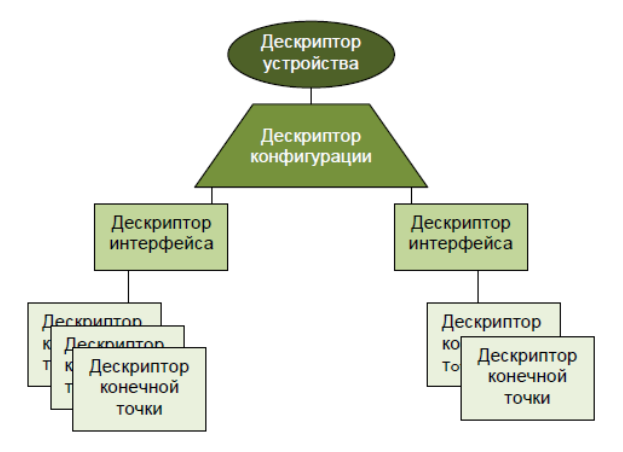
ОС Windows обеспечивает программную поддержку USB-устройств через стек стандартных драйверов, которые выполняют функции по управлению устройствами и обмену данными между ними и системой. Программирование взаимодействия с USB-устройствами предполагает написание собственной программы-драйвера, что достаточно трудоемко.

Для упрощения работы с USB-устройствами из прикладных программ была создана многоплатформенная библиотека libusb. Она реализует универсальный низкоуровневый интерфейс, который позволяет взаимодействовать со всеми возможными USB-устройствами. С помощью библиотеки libusb можно получить список доступных USB-устройств, узнать их параметры, обмениваться данными с устройствами в синхронном и асинхронном режимах, реагировать на подключение и отключение устройств.

**Дескрипторы USB-устройства**

В этой структуре, формат которой строго регламентируется стандартом, должна храниться вся информация, с помощью которой хост может автоматически конфигурироваться и начинать нормальную работу с USB-устройством. Именно набором дескрипторов USB-устройства обменивается с хостом во время процесса энумерации.

Имеется 4 основных типа дескрипторов: устройства, конфигурации, интерфейса, конечных точек. Все эти типы дескрипторов должны обязательно присутствовать в USB-устройстве.



*Рис. 1. Типы дескрипторов.*

Конечные точки объединяются в интерфейс. Каждому интерфейсу соответствует драйвер операционной системы (ОС) хоста. Некоторые устройства могут иметь несколько интерфейсов, которые могут функционировать одновременно. Например, телефонный аппарат может иметь интерфейс клавиатуры и аудио-интерфейс, тогда ОС хоста будет использовать два различных драйвера для каждого интерфейса, а прикладная программа будет взаимодействовать с этими драйверами.

Несколько интерфейсов объединяются в конфигурацию. У USB устройства может быть несколько конфигураций, но не могут быть активными несколько конфигураций одновременно; возможно переключение между конфигурациями.

**Выполнение работы:**

Работа выполнялась на ПК с установленной Windows 11 в редакторе VisualStudio Code, с последующим тестированием на Windows 10 в среде разработки VisualStudio 2017. Для реализации программы была использована библиотека для работы с USB-устройствами Libusb, программирование велось на языке C++. Помимо самой библиотеки, на USB-устройства, с которыми осуществлялась работа, был установлен фильтр-драйвер под архитектуру amd64 (путь к фильтру, который находится в одном архиве с библиотекой: libusb-win32-bin-1.2.6.0.zip\libusb-win32-bin-1.2.6.0\bin\amd64), подробная инструкция по установке фильтра приведена в методичке к лабораторной работе.

Алгоритм работы программы:

Основной алгоритм (функция *main()*):

Начало

1. Запуск бесконечного цикла, для динамического обновления информации об активных USB-устройствах. Как только прерывается выполнение цикла переход к п. 17.
2. Инициализация libusb (всех структур для работы с USB-устройством).
3. Поиск всех USB-шин в системе.
4. Поиск всех USB-устройств на каждой из найденных шин.
5. Найдена ли другая шина USB? Если нет, то переход к п.9.
6. Создаем элемент структуры USB-устройства.
7. Есть ли USB-устройство на текущей шине? Если нет, то переход к п.5.
8. Вызываем подпрограмму (*print\_device()*), в качестве аргумента передаем структуру USB-устройства.
9. Проверяем изменилось ли количество устройств. Если нет, то переход к п. 1.
10. Инициализация libusb (всех структур для работы с USB-устройством).
11. Поиск всех USB-шин в системе.
12. Поиск всех USB-устройств на каждой из найденных шин.
13. Найдена ли другая шина USB? Если нет, то переход к п. 9.
14. Создаем элемент структуры USB-устройства.
15. Есть ли USB-устройство на текущей шине? Если нет, то переход к п. 13.
16. Увеличиваем счетчик для подсчета устройств на 1.
17. Завершение работы программы.

Конец

Подпрограмма (функция *print\_device()*):

Начало

1. Открываем найденное USB-устройство.
2. Открылось ли устройство? Если да, то переход к п. 5.
3. Выводим сообщение, что устройство открыть не удалось и переходим к п. 15.
4. Выводим дескрипторы VID, PID.
5. Извлекаем строковые дескрипторы о: производителе, продукте, серийном номере.
6. Выводим полученные дескрипторы.
7. Вызываем подпрограмму (*print\_device\_desc()*), в качестве аргумента передаем ссылку не дескриптор USB-устройства.
8. Значение счетчика меньше количества конфигураций, поддерживаемых устройством? Если нет, то переход к п. 11.
9. Вызываем подпрограмму (*print\_config\_desc()*), в качестве аргумента передаем ссылку на дескриптор конфигурации устройства.
10. Значение счетчика меньше количества интерфейсов поддерживаемых конфигурацией? Если нет, то переход к п. 13.
11. Вызываем подпрограмму (*print\_interface\_desc()*), в качестве аргумента передаем ссылку на дескриптор интерфейса устройства.
12. Значение счетчика меньше числа конечных точек устройства? Если нет, то переход к п. 15.
13. Вызываем подпрограмму (*print\_endpoint\_desc()*), в качестве аргумента передаем ссылку на дескриптор конечной точки устройства.
14. Выход из подпрограммы и возврат к основной программе.

Конец

Подпрограмма (функция *print\_device\_desc()*):

Начало

1. Выводим дескрипторы непосредственно USB-устройств.

Конец

Подпрограмма (функция *print\_config\_desc()*):

Начало

1. Выводим дескрипторы конфигурации USB-устройств.

Конец

Подпрограмма (функция *print\_interface\_desc()*):

Начало

1. Выводим дескрипторы интерфейса USB-устройств.

Конец

Подпрограмма (функция *print\_endpoint\_desc()*):

Начало

1. Выводим дескрипторы конечных точек USB-устройств.

Конец

При компиляции проекта в среде VisualStudio 2017 был использован файл с библиотекой libusb.lib, находящийся по пути: libusb-win32-bin-1.2.6.0.zip\libusb-win32-bin-1.2.6.0\lib\msvc\_x64. Поскольку, при использовании файла из папок, находящихся по пути:libusb-win32-bin-1.2.6.0.zip\libusb-win32-bin-1.2.6.0\lib\gcc, а также libusb-win32-bin-1.2.6.0.zip\libusb-win32-bin-1.2.6.0\lib\msvc, при сборке появлялась ошибка, причиной которой может быть, например, повреждение библиотеки, которая использовалась при компиляции.

**Результаты работы:**

Результатом выполнения лабораторной работы стала программа, получающая список подключенных к машине USB-устройств и вывод его дескрипторов на экран, а также, динамически обновляющая информацию об активных USB-устройствах при их подключении и отключении.

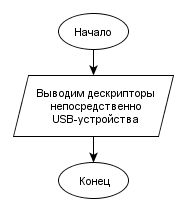
Блок-схема программы:



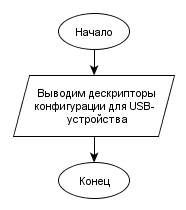
*Рис. 2. Функция main().*

**

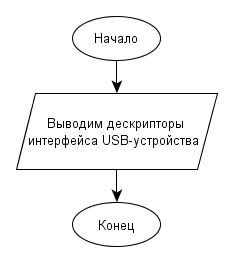
*Рис. 3. Функция print\_device().*

**

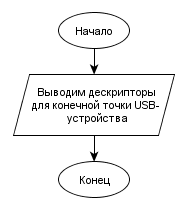
*Рис.4. Функция print\_device\_desc().*

**

*Рис. 5. Функция print\_config\_desc().*

**

*Рис.6. Функция print\_interface\_desc().*

**

*Рис. 7. Функция print\_endpoint\_desc().*

Листинг программы:

#include <lusb0\_usb.h>

#include <iostream>

#include <iomanip>

#include <stdio.h>

#include <conio.h>

#include <Windows.h>

#include <stdlib.h>

using namespace std;

print\_device\_desc(&dev->descriptor);

    for (inti = 0; i<int(dev->descriptor.bNumConfigurations); i++)

    {

        print\_config\_desc(&dev->config[i]);

    }

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

    {

        print\_interface\_desc(&dev->config->interface->altsetting[i]);

    }

    for (int i = 0; i<int(dev->config->interface->altsetting->bNumEndpoints); i++)

    {

        print\_endpoint\_desc(&dev->config->interface->altsetting->endpoint[i]);

    }

    return;

}

voidprint\_device\_desc(structusb\_device\_descriptor\*device)

{

    cout<<left<<setw(24) <<dec<<"  bLength: "<<int(device->bLength) <<endl;

    cout<<left<<hex<<setw(24) <<"  bDescriptorType: "<<int(device->bDescriptorType) <<"h"<<endl;

    cout<<left<<hex<<setw(24) <<"  bcdUSB: "<<int(device->bcdUSB) <<"h"<<endl;

    cout<<left<<hex<<setw(24) <<"  bDeviceClass: "<<int(device->bDeviceClass) <<"h"<<endl;

    cout<<left<<hex<<setw(24) <<"  bDeviceSubClass: "<<int(device->bDeviceSubClass) <<"h"<<endl;

    cout<<left<<hex<<setw(24) <<"  bDeviceProtocol: "<<int(device->bDeviceProtocol) <<"h"<<endl;

    cout<<left<<hex<<setw(24) <<"  bMaxPacketSize0: "<<int(device->bMaxPacketSize0) <<"h"<<endl;

    cout<<left<<hex<<setw(24) <<"  idVendor: "<<int(device->idVendor) <<"h"<<endl;

    cout<<left<<hex<<setw(24) <<"  idProduct: "<<int(device->idProduct) <<"h"<<endl;

    cout<<left<<hex<<setw(24) <<"  bcdDevice: "<<int(device->bcdDevice) <<"h"<<endl;

    return;

}

voidprint\_config\_desc(structusb\_config\_descriptor\*config)

{

    cout<<left<<setw(27) <<"    wTotalLength: "<<config->wTotalLength<<endl;

    cout<<left<<setw(27) <<"    bNumInterfaces: "<<int(config->bNumInterfaces) <<endl;

    cout<<left<<setw(27) <<"    bConfigurationValue: "<<int(config->bConfigurationValue) <<endl;

    cout<<left<<setw(27) <<"    iConfiguration: "<<int(config->iConfiguration) <<endl;

    cout<<left<<setw(27) <<hex<<"    bmAttributes: "<<int(config->bmAttributes) <<"h"<<endl;

    cout<<left<<setw(27) <<"    MaxPower: "<<int(config->MaxPower) <<endl;

    return;

}

voidprint\_interface\_desc(structusb\_interface\_descriptor\*interface)

{

    cout<<left<<setw(30) <<"       bInterfaceNumber: "<<int(interface->bInterfaceNumber) <<endl;

    cout<<left<<setw(30) <<"       bAlternateSetting: "<<int(interface->bAlternateSetting) <<endl;

    cout<<left<<setw(30) <<"       bNumEndpoints: "<<int(interface->bNumEndpoints) <<endl;

    cout<<left<<setw(30) <<"       bInterfaceClass: "<<int(interface->bInterfaceClass) <<endl;

    cout<<left<<setw(30) <<"       bInterfaceSubClass: "<<int(interface->bInterfaceSubClass) <<endl;

    cout<<left<<setw(30) <<"       bInterfaceProtocol: "<<int(interface->bInterfaceProtocol) <<endl;

    cout<<left<<setw(30) <<"       iInterface: "<<int(interface->iInterface) <<endl;

    return;

}

voidprint\_endpoint\_desc(structusb\_endpoint\_descriptor\*endpoint)

{

    cout<<left<<setw(33) <<hex<<"          bEndpointAddress: "<<int(endpoint->bEndpointAddress) <<"h"<<endl;

    cout<<left<<setw(33) <<hex<<"          bmAttributes: "<<int(endpoint->bmAttributes) <<"h"<<endl;

    cout<<left<<setw(33) <<"          wMaxPacketSize: "<<int(endpoint->wMaxPacketSize) <<endl;

    cout<<left<<setw(33) <<"          bInterval: "<<int(endpoint->bInterval) <<endl;

    cout<<left<<setw(33) <<"          bRefresh: "<<int(endpoint->bRefresh) <<endl;

    cout<<left<<setw(33) <<"          bSynchAddress: "<<int(endpoint->bSynchAddress) <<endl<<endl;

    return;

}

voidprint\_device(structusb\_device\*dev)

{

    usb\_dev\_handle \*udev;

    charstring[256];

    udev = usb\_open(dev);

    if (udev == 0)

    {

        cout<<"Невозможно открыть USB-устройство "<<usb\_strerror()<<endl;

    return;

    }

    cout<<hex<<uppercase<<"Device VID: "<<dev->descriptor.idVendor<<endl;

    cout<<hex<<uppercase<<"Device PID: "<<dev->descriptor.idProduct<<endl;

    usb\_get\_string\_simple(udev, dev->descriptor.iManufacturer, string, 256);

    cout<<"Manufacturer: "<<string<<endl;

    usb\_get\_string\_simple(udev, dev->descriptor.iProduct, string, 256);

    cout<<"Product: "<<string<<endl;

    usb\_get\_string\_simple(udev, dev->descriptor.iSerialNumber, string, sizeof(string));

    cout<<"Serial Number: "<<string<<endl;

int main()

{

    while (true)

    {

        int chet = 0;

        int chet2 = 0;

        usb\_init();

        usb\_find\_busses();

        usb\_find\_devices();

        usb\_bus \*bus;

        for (bus = usb\_busses; bus; bus = bus->next)

        {

            struct usb\_device \*dev;

            for (dev = bus->devices; dev; dev = dev->next)

            {

                chet++;

                print\_device(dev);

                cout<<"/------------------------------------------/\n"<<endl;

            }

        }

        chet2 = chet;

        while (chet == chet2)

        {

            chet2 = 0;

            usb\_init();

            usb\_find\_busses();

            usb\_find\_devices();

            usb\_bus \*bus;

            for (bus = usb\_busses; bus; bus = bus->next)

            {

                struct usb\_device \*dev;

                for (dev = bus->devices; dev; dev = dev->next)

                {

                    chet2++;

                }

            }

        }

        system("cls");

    }

    system("pause");

    return0;

}

После запуска программы на выполнение, результат ее работы выглядит следующим образом:

Device VID: 10C4 ID Vendor

Device PID: 8105 ID Product

Manufacturer: YSPRINGTECH

Product: USB OPTICAL MOUSE

Serial Number: ?

bLength: 18 Размер дескриптора в байтах

bDescriptorType: 1h Тип дескриптора

bcdUSB: 200h Номер спецификации USB, с которой совместимо устройство

bDeviceClass: 0h Код класса

bDeviceSubClass: 0h Код подкласса

bDeviceProtocol: 0h Код протокола

bMaxPacketSize0: 8h Максимальный размер пакета для конечной точки 0

idVendor: 10C4h ID Vendor

idProduct: 8105h ID Product

bcdDevice: 0h Номер версии устройства

wTotalLength: 22 Полная длина возвращаемых данных в байтах

bNumInterfaces: 1 Количество интерфейсов

bConfigurationValue: 1 Величина, используемая как аргумент для выбора этой конфигурации

iConfiguration: 0 Индекс строки, описывающей эту конфигурацию

bmAttributes: A0h Задает параметры питания для конфигурации

MaxPower: 32 Максимальное энергопотребление

bInterfaceNumber: 0 Порядковый номер интерфейса

bAlternateSetting: 0 Величина, используемая для выбора альтернативной установки

bNumEndpoints: 1 Количество конечных точек, используемых в интерфейсе

bInterfaceClass: 3 Код класса

bInterfaceSubClass: 1 Код подкласса

bInterfaceProtocol: 2 Код протокола

iInterface: 0 Индекс строки, описывающей этот интерфейс

bEndpointAddress: 81h Адрес конечной точки

bmAttributes: 3h Указывает тип передачи

wMaxPacketSize: 8 Максимальный размер пакета этой конечной точки, подходящий для отправки или приема

bInterval: A Интервал для того, чтобы опросить передачи данных конечной точки

bRefresh: 0 скорость, с которой обеспечивается обратная связь синхронизации для аудио устройств

bSynchAddress: 0 Адрес конечной точки для отправки информации о синхронизации

/-------------------------------------------/

*Рис. 8. Результат работы программы.*

**Вывод:** В ходе выполнения лабораторной работы было осуществлено знакомство с библиотекой Libusb, с интерфейсом USB, получены навыки программирования периферийных USB-устройств с использованием библиотеки Libusb. Была реализована программа, получающая список подключенных к машине USB-устройств, для каждого найденного устройства напечатаны все его дескрипторы. В программу была добавлена возможность динамического обновления информации об активных USB-устройствах при их подключении и отключении.