МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №6
по курсу «ЭВМ и периферийные устройства»

НИЗКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ

Выполнил: студент 2-го курса гр. 17208
Гафиятуллин А.Р.

Новосибирск, 2018

1. ЦЕЛИ РАБОТЫ:

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

2. ХОД РАБОТЫ:

1. На языке программирования C++ была реализована программа, получающая список всех подключенных к машине USB устройств с использованием libusb. Для каждого найденного устройства создавался временный объект класса Device, который печатал его класс, идентификатор производителя, идентификатор изделия и серийный номер.
2. Исходный код реализованной программы:

```
#include <iostream>
#include <libusb.h>
#include <iomanip>
using namespace std;

class Device {
private:
    libusb_device* dev_;
    libusb_device_descriptor desc_;
    libusb_config_descriptor* config_;
public:
    Device(libusb_device* dev);

    static string getDeviceClass(int deviceClass);

    string getInterfacesInfo() const;

    friend ostream& operator<<(ostream& s, const Device& dev);
```

```
    string getSerialNumber() const;

    ~Device();
};

Device::Device(libusb_device* dev){
    dev_ = dev;
    int errorCode = libusb_get_device_descriptor(dev_, &desc_);
    if(errorCode < 0)
        throw domain_error("Error: can't get a device descriptor, code: "
            + to_string(errorCode));
    libusb_get_config_descriptor(dev_, 0, &config_);
}

string Device::getDeviceClass(int deviceClass){
    switch(deviceClass){
    case LIBUSB_CLASS_PER_INTERFACE :
        return string("Each interface specifies its own class information");
    case LIBUSB_CLASS_AUDIO :
        return string("Audio class");
    case LIBUSB_CLASS_COMM :
        return string("Communications class");
    case LIBUSB_CLASS_HID :
        return string("Human Interface Device class");
    case LIBUSB_CLASS_PHYSICAL :
        return string("Physical");
    case LIBUSB_CLASS_PTP :
        return string("Image class");
    case LIBUSB_CLASS_PRINTER :
        return string("Printer class");
    case LIBUSB_CLASS_MASS_STORAGE :
        return string("Mass storage class");
    case LIBUSB_CLASS_HUB :
        return string("Hub class");
```

```
    case LIBUSB_CLASS_DATA :
        return string("Data class");
    case LIBUSB_CLASS_SMART_CARD :
        return string("Smart Card");
    case LIBUSB_CLASS_CONTENT_SECURITY :
        return string("Content Security");
    case LIBUSB_CLASS_VIDEO :
        return string("Video");
    case LIBUSB_CLASS_PERSONAL_HEALTHCARE :
        return string("Personal Healthcare");
    case LIBUSB_CLASS_DIAGNOSTIC_DEVICE :
        return string("Diagnostic Device");
    case LIBUSB_CLASS_WIRELESS  :
        return string("Wireless class");
    case LIBUSB_CLASS_APPLICATION :
        return string("Application class");
    case LIBUSB_CLASS_VENDOR_SPEC :
        return string("Class is vendor-specific");
    default :
        return string(" ");
    }
}

string Device::getInterfacesInfo() const {
    const libusb_interface* inter;
    const libusb_interface_descriptor* interdesc;
    stringstream interfacesInfo;
    interfacesInfo << right << setw(30) << "Number of interfaces: "
            << to_string(config_->bNumInterfaces) << endl;
    for(int i = 0; i < config_->bNumInterfaces; i++){
        inter = &config_->interface[i];
        interfacesInfo << right << setw(15) << i + 1 << ") "
            << "Number of alternate settings:  "
            << inter->num_altsetting << endl;
```

```
        for(int j = 0; j < inter->num_altsetting; j++){
            interdesc = &inter->altsetting[j];
            interfacesInfo << right << setw(20) << j + 1 << ") "
                << "Interface class: "
                << getDeviceClass(interdesc->bInterfaceClass) << endl;
        }
    }
    return interfacesInfo.str();
}

string Device::getSerialNumber() const {
    libusb_device_handle* handle;
    int errorCode = libusb_open(dev_, &handle);
    if(errorCode < 0)
        throw domain_error("Error: can't open device, code: " +
            to_string(errorCode));
    unsigned char serialNumber[255];
    libusb_get_string_descriptor_ascii(handle, desc_.iSerialNumber,
        serialNumber, 255);
    libusb_close(handle);
    return string(reinterpret_cast<char*>(serialNumber));
}

ostream& operator << (ostream& s, const Device& dev){
    s << left << setw(15) << setfill(' ') << hex
        << "Device class: " << Device::getDeviceClass(dev.desc_.bDeviceClass)
            << endl << left << setfill(' ') << setw(15)
        << "Serial number: " << dev.getSerialNumber() << endl << left
            << setfill(' ') << setw(15)
        << "Vendor ID: " << right << setfill('0') << setw(4)
            << dev.desc_.idVendor << endl << left << setfill(' ') << setw(15)
        << "Product ID: " << right << setw(4) << setfill('0')
            << dev.desc_.idProduct << endl << right << setfill(' ') <<
setw(28)
```

```
        << "Interfaces information: " << endl << dev.getInterfacesInfo() <<
endl;
    return s;
}

Device::~Device(){
        libusb_free_config_descriptor(config_);
}

int main(){
    libusb_device** devs;
    libusb_context* ctx = nullptr; //session context
    int errorCode;
    ssize_t cnt;        //amount of devices
    errorCode = libusb_init(&ctx);
    if(errorCode < 0)
        throw domain_error("Error: can't initialize session, code: " +
            to_string(errorCode));
    libusb_set_debug(ctx, 3);
    cnt = libusb_get_device_list(ctx, &devs);
    if(cnt < 0)
        throw domain_error("Error: can't get device list, code: " +
            to_string(errorCode));
    for(ssize_t i = 0; i < cnt; i++){
        cout << i + 1 << ") " << endl << Device(devs[i]) << endl;
    }
    libusb_free_device_list(devs, 1);
    libusb_exit(ctx);
    return 0;
}
```

3. Команда компиляции:

   g++ -std=c++17 -o usb_inf.out -I/usr/include/libusb-1.0 Device.cpp -lusb-1.0

4. Описание обнаруженных USB-устройств:

4.1. 2 корневых USB-концентратора от Linux Foundation с 1 настройкой на 1 интерфейсе:

Device class: Hub class

Serial number: 0000:00:14.0

Vendor ID: 1d6b

Product ID: 0003

Interfaces information:

Number of interfaces: 1

1) Number of alternate settings: 1

1) Interface class: Hub class

и

Device class: Hub class

Serial number: 0000:00:14.0

Vendor ID: 1d6b

Product ID: 0002

Interfaces information:

Number of interfaces: 1

1) Number of alternate settings: 1

1) Interface class: Hub class

4.2. Картридер от Realtek с 1 настройкой на 1 интерфейсе:

Device class: Class is vendor-specific

Serial number: 20100201396000000

Vendor ID: 0bda

Product ID: 0129

Interfaces information:

Number of interfaces: 1

1) Number of alternate settings: 1

1) Interface class: Class is vendor-specific

4.3. VGA веб-камера с 2 интерфейсами по 1 и 8 настроек:

Device class:

Serial number: 200901010001

Vendor ID: 0bda

Product ID: 57b3

Interfaces information:

Number of interfaces: 2

1) Number of alternate settings: 1

1) Interface class: Video

2) Number of alternate settings: 8

1) Interface class: Video

2) Interface class: Video

3) Interface class: Video

4) Interface class: Video

5) Interface class: Video

6) Interface class: Video

7) Interface class: Video

8) Interface class: Video

4.4. Bluetooth от Intel с 2 интерфейсами по 1 и 6 настроек:

Device class: Wireless class

Serial number:

Vendor ID: 8087

Product ID: 0aa7

Interfaces information:

Number of interfaces: 2

1) Number of alternate settings: 1

1) Interface class: Wireless class

2) Number of alternate settings: 6

1) Interface class: Wireless class

2) Interface class: Wireless class

3) Interface class: Wireless class

4) Interface class: Wireless class

5) Interface class: Wireless class

6) Interface class: Wireless class

4.5. Беспроводная мышь от Xiaomi с 1 настройкой на 1 интерфейсе:

Device class: Each interface specifies its own class information

Serial number:

Vendor ID: 2717

Product ID: 5001

Interfaces information:

Number of interfaces: 1

1) Number of alternate settings: 1

1) Interface class: Human Interface Device class

3. ВЫВОДЫ:

1. Ознакомились с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb;
2. USB-устройства имеют сложную иерархию дескрипторов, которые предоставляют различную информацию о своем уровне.