

ЛАБОРАТОРНАЯ РАБОТА №6

НИЗКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ

Цели работы

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

1. БИБЛИОТЕКА LIBUSB

USB (Universal Serial Bus) – последовательный интерфейс передачи данных для среднескоростных и низкоскоростных периферийных устройств. Стандарт USB ввел единообразие в работу с широким спектром периферийных устройств. До его появления использовалось множество разных интерфейсов для подключения периферийных устройств (последовательный порт RS-232C для модема, мыши, параллельный порт для принтера и дисководов IOMEGA ZIP, PS/2 для мыши и клавиатуры, специализированные интерфейсы для подключения сканеров и т. д.).

Программирование взаимодействия с USB-устройствами 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.

Каждое конкретное устройство имеет свой специальный протокол взаимодействия, и прикладная программа должна взаимодействовать с

устройством в соответствии с его протоколом, используя низкоуровневые операции библиотеки libusb.

2. ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ДОСТУПНЫХ USB-УСТРОЙСТВАХ С ПОМОЩЬЮ БИБЛИОТЕКИ LIBUSB

Библиотека libusb предоставляет следующие функции, которые позволяют получить информацию об имеющихся в системе USB-устройствах:

- libusb_init – инициализация работы с libusb,
- libusb_exit – завершение работы с libusb,
- libusb_set_debug – установка уровня подробности отладочных сообщений (рекомендуется использовать уровень 3),
- libusb_get_device_list – получение списка подключенных к машине USB устройств,
- libusb_free_device_list – освобождение памяти, выделенной в функции libusb_get_device_list для хранения данных со списком устройств,
- libusb_get_device_descriptor – получение дескриптора USB устройства,
- libusb_get_config_descriptor – получение дескриптора конфигурации USB устройства,
- libusb_free_config_descriptor – освобождение памяти, выделенной в функции libusb_get_config_descriptor,
- libusb_ref_device – увеличение счетчика числа пользователей устройства на 1 (при первом вызове функции libusb_get_device_list после подключения устройства его счетчик устанавливается в 1),
- libusb_unref_device – уменьшение счетчика числа пользователей устройства на 1 (если счетчик уменьшается до нуля, дескриптор устройства удаляется),

- `libusb_open` – открыть устройство (начать работать с устройством) и получить дескриптор устройства, который далее можно использовать для ввода/вывода данных,
- `libusb_open_device_with_vid_pid` – открыть устройство по его идентификаторам производителя и изделия,
- `libusb_close` – закрыть устройство после его использования,
- `libusb_get_string_descriptor_ascii` – получить дескриптор устройства в виде строки символов,
- `lib_usb_error_name` – преобразование кода ошибки библиотеки `libusb` в строковое сообщение об ошибке.

Пример программы получения состава и параметров USB-устройств с помощью библиотеки `libusb` приведен в листинге 1.

Листинг 1. Получение параметров USB-устройств

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

using namespace std;

void printdev(libusb_device *dev);

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
    // инициализировать библиотеку libusb, открыть сессию работы с libusb
    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_debug(ctx, 3);
    // получить список всех найденных USB- устройств
```

```

cnt = libusb_get_device_list(ctx, &devs);
if(cnt < 0){
    fprintf(stderr,
        "Ошибка: список USB устройств не получен.\n", r);
    return 1;
}
printf("найдено устройств: %d\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("+---+---+---+---+---+---+---+---+---+\n");
printf("--+---+-----\n");
for(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);
libusb_exit(ctx);          // завершить работу с библиотекой libusb,
                           // закрыть сессию работы с libusb
return 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,
            "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++){
        inter = &config->interface[i];
        printf("| | | | | "
            "%.2d %.2d | | | |\n",
            inter->num_altsetting,
            (int)desc.bDeviceClass
        );
        for(int j=0; j<inter->num_altsetting; j++) {
            interdesc = &inter->altsetting[j];
            printf("| | | | | | | | "
                "%.2d %.2d | | |\n",
                (int)interdesc->bInterfaceNumber,
                (int)interdesc->bNumEndpoints
            );
            for(int k=0; k<(int)interdesc->bNumEndpoints; k++) {
                epdesc = &interdesc->endpoint[k];
                printf(
                    "| | | | | | | | | | "
                    "%.2d %.9d\n",
                    (int)epdesc->bDescriptorType,
                    (int)(int)epdesc->bEndpointAddress
                );
            }
        }
    }
    libusb_free_config_descriptor(config);
}

```

Для компиляции программы в листинге 1 можно использовать команду:

```
g++ -o executable -I/usr/include/libusb-1.0 main.cpp -lusb-1.0
```

Примечание. При запуске на собственном компьютере может понадобиться разрешить доступ к USB-устройствам по чтению и записи. Например:

```
chmod 777 /dev/bus/usb/001/003
```

Альтернативный вариант заключается в запуске программы с правами администратора:

```
sudo ./executable
```

3. ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ

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. 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.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое libusb? Какие задачи решает libusb?
2. На какие группы можно разбить функции libusb?
3. Как с помощью библиотеки libusb получить состав и конфигурацию USB-устройств?

5. ЛИТЕРАТУРА

1. [https://github.com/forthnutter/USB/blob/master/doc/Introduction To Using LibUSB-1.0 - C++ Tutorials _ Dream.In.pdf](https://github.com/forthnutter/USB/blob/master/doc/Introduction%20To%20Using%20LibUSB-1.0-C%2B%2BTutorials_Dream.In.pdf)
2. <https://libusb.info/>

ПРИЛОЖЕНИЕ

Коды классов USB устройств

00h – код отсутствует (информацию о классе нужно получать в дескрипторе интерфейса)

01h – аудиоустройство (если код получен из дескриптора интерфейса, а не устройства)

02h – коммуникационное устройство (сетевой адаптер)

03h – устройство пользовательского интерфейса

05h – физическое устройство

06h – изображения

07h – принтер

08h – устройство хранения данных

09h – концентратор

0Ah – CDC-Data

0Bh – Smart Card

0Dh – Content Security

0Eh – видеоустройство

0Fh – персональное медицинское устройство

10h – аудио- и видеоустройства

DCh – диагностическое устройство

E0h – беспроводный контроллер

EFh – различные устройства

FE – специфическое устройство