Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Дисциплина: Интерфейсы и устройства вычислительных машин

Лабораторная работа №5

«USB-устройства»

Выполнила: Проверил:

Ст. гр. 150504 Селезнев А. И.

Горбачевский К. В.

Минск, 2023

1. Постановка задачи

Реализовать мониторинг USB-портов:

- отслеживать появление нового USB-устройства в системе;

- события безопасного и небезопасного извлечения устройства в консоли;

- отказ в безопасном извлечении.

Для всех извлекаемых USB-устройств предусмотреть возможность программного безопасного извлечения. Приложение должно корректно работать с модифицированными USB устройствами – например, разбитыми на пару дисков или смонтированными как CD-ROM + Flash.

Список устройств подключение которых необходимо продемонстрировать:

- мышь;

- флэшка.

1. Краткие теоретические сведения

**USB** (Universal serial bus) — универсальный стандарт подключения к ПК и другим электронным устройствам различного периферийного оборудования.

**Область применения USB**: устройства ввода, принтеры, сканеры, аудиоустройства, фото-/видеокамеры, коммуникации (модемы), устройства хранениия, игровые устройства, телефоны, мониторы, электронные ключи, HID-устройства

Типы USB коннекторов:

**USB Type-A:** это наиболее распространенный и широко используемый коннектор. Он часто используется для подключения устройств (клавиатуры, флешки, мыши) к компьютерам, ноутбукам, зарядных устройствам и другим устройствам.

**USB Type-B:** этот коннектор чаще всего используется для подключения принтеров, сканеров и некоторых старых устройств к компьютерам.

**Micro USB:** этот коннектор меньше по размеру и часто используется для зарядки мобильных устройств, таких как смартфоны и планшеты. Также его можно найти на некоторых камерах и других портативных устройствах.

**Mini USB:** этот коннектор имеет размер промежуточный между Micro USB и стандартным USB Type-B. Он часто встречается в старых камерах, MP3-плеерах и других портативных устройствах.

**USB Type-C:** это наиболее актуальный и перспективный разъем, полностью симметричный и двухсторонний. Появился одновременно со стандартом USB 3.1 и актуален для более поздних версий стандартов USB. Он используется в современных смартфонах, ноутбуках, планшетах, внешних накопителях и других устройствах.

1. Код программы

#include <windows.h>

#include <iostream>

#include <setupapi.h>

#include <initguid.h>

#include <Usbiodef.h>

#include <clocale>

#include <thread>

#include <functional>

#include <vector>

#include <map>

#include <chrono>

#include <Cfgmgr32.h>

#pragma comment(lib, "setupapi.lib")

using namespace std;

vector<wstring> devices;

map<wstring, DWORD> devicesINST;

CONFIGRET EjectVolume(DEVINST devInst)

{

CONFIGRET result = CM\_Request\_Device\_EjectW(devInst, nullptr, nullptr, NULL, NULL);

if (result != CR\_SUCCESS) {

std::wcout << L"Error: ejecting denied." << endl;

}

return result;

}

int printUSBDevicesInfo(bool isPrint = false, bool isInitial = false)

{

SP\_DEVINFO\_DATA DeviceInfoData;

ZeroMemory(&DeviceInfoData, sizeof(SP\_DEVINFO\_DATA));

DeviceInfoData.cbSize = sizeof(SP\_DEVINFO\_DATA);

HDEVINFO DeviceInfoSet = SetupDiGetClassDevs(&GUID\_DEVINTERFACE\_USB\_DEVICE, NULL, NULL, DIGCF\_PRESENT | DIGCF\_DEVICEINTERFACE);

if (DeviceInfoSet == INVALID\_HANDLE\_VALUE) {

std::wcout << L"Error: Unable to get device information set." << endl;

return NULL;

}

if (isInitial && devices.size()) {

devices.clear();

devicesINST.clear();

}

int i = 0;

for (; ; i++)

{

DeviceInfoData.cbSize = sizeof(SP\_DEVINFO\_DATA);

if (!SetupDiEnumDeviceInfo(DeviceInfoSet, i, &DeviceInfoData))

{

break;

}

const int PropertyBufferSize = 1024;

wchar\_t deviceID[PropertyBufferSize], deviceName[PropertyBufferSize], companyName[PropertyBufferSize];

ZeroMemory(&deviceID, sizeof(deviceID));

ZeroMemory(&deviceName, sizeof(deviceName));

ZeroMemory(&companyName, sizeof(companyName));

if (!SetupDiGetDeviceInstanceId(DeviceInfoSet, &DeviceInfoData, deviceID, sizeof(deviceID), NULL))

{

std::wcout << L"Error: Unable to get device instance ID." << endl;

std::wcout << L"Error code: " << GetLastError() << endl;

SetupDiDestroyDeviceInfoList(DeviceInfoSet);

return NULL;

}

if (!SetupDiGetDeviceRegistryProperty(DeviceInfoSet, &DeviceInfoData, SPDRP\_DEVICEDESC, NULL, (PBYTE)deviceName, sizeof(deviceName), NULL))

{

std::wcout << L"Error: Unable to get device description." << endl;

SetupDiDestroyDeviceInfoList(DeviceInfoSet);

return NULL;

}

if (!SetupDiGetDeviceRegistryProperty(DeviceInfoSet, &DeviceInfoData, SPDRP\_MFG, NULL, (PBYTE)companyName, sizeof(companyName), NULL))

{

std::wcout << L"Error: Unable to get manufacturer information." << endl;

SetupDiDestroyDeviceInfoList(DeviceInfoSet);

return NULL;

}

wstring venAndDevId(deviceID);

if (isPrint) {

std::wcout << L"\n\t\t\tName: " << deviceName << endl;

std::wcout << L"\t\t\tVendor ID: " << venAndDevId.substr(8, 4).c\_str() << endl;

std::wcout << L"\t\t\tDevice ID: " << venAndDevId.substr(17, 4).c\_str() << endl << endl;

}

if (isInitial) {

wstring devId = venAndDevId.substr(17, 4);

devices.push\_back(deviceName);

devicesINST[devId] = DeviceInfoData.DevInst;

}

}

SetupDiDestroyDeviceInfoList(DeviceInfoSet);

return i;

}

int findElement(vector<wstring> v, wstring value) {

auto it = find(v.begin(), v.end(), value);

if (it != v.end()) {

int index = it - v.begin();

return index;

}

else return -1;

}

wstring findConnectedDevice(vector<wstring> tmp) {

for (int i = 0; i < devices.size(); i++) {

int index = findElement(tmp, devices[i]);

if (index == -1) {

return devices[i];

}

}

}

wstring findDisconnectedDevice(vector<wstring> tmp) {

for (int i = 0; i < tmp.size(); i++) {

int index = findElement(devices, tmp[i]);

if (index == -1) {

return tmp[i];

}

}

}

void checkDevices() {

int devicesCount = printUSBDevicesInfo(false, true);

while (1) {

devicesCount = printUSBDevicesInfo(false, false);

if (devicesCount > devices.size()) {

vector<wstring> tmp = devices;

printUSBDevicesInfo(false, true);

wstring connectedDevice = findConnectedDevice(tmp);

wcout << "\nDevice " << connectedDevice << " connected\n";

}

else if (devicesCount < devices.size()) {

vector<wstring> tmp = devices;

wstring disconnectedDevice;

printUSBDevicesInfo(false, true);

disconnectedDevice = findDisconnectedDevice(tmp);

wcout << "\nDevice " << disconnectedDevice << " disconnected\n";

}

}

}

void interval(std::function<void(void)> func, unsigned int interval) {

std::thread([func, interval]()

{

while (true)

{

auto x = std::chrono::steady\_clock::now() + std::chrono::milliseconds(interval);

func();

std::this\_thread::sleep\_until(x);

}

}).detach();

}

void localization() {

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

}

void ejectDevice() {

string deviceId;

cout << "\nEnter deviceId that should be unconnect: ";

cin >> deviceId;

CONFIGRET result = EjectVolume(devicesINST[wstring(deviceId.begin(), deviceId.end())]);

}

void menu() {

char key = '0';

cout << "\n\t\t\t================== Menu ==================\n";

cout << "\n\t\t\t[i] - Print connected devices information";

cout << "\n\t\t\t[e] - Eject connected device\n";

cout << "\n\t\t\t==========================================\n\n";

while (1) {

cout << "\n\nEnter action (i/e): ";

cin >> key;

switch (key) {

case 'i': printUSBDevicesInfo(true, false); break;

case 'e': ejectDevice(); break;

default: break;

}

}

}

int main() {

localization();

interval(checkDevices, 500);

menu();

return 0;

}

1. **Тестирование программы**

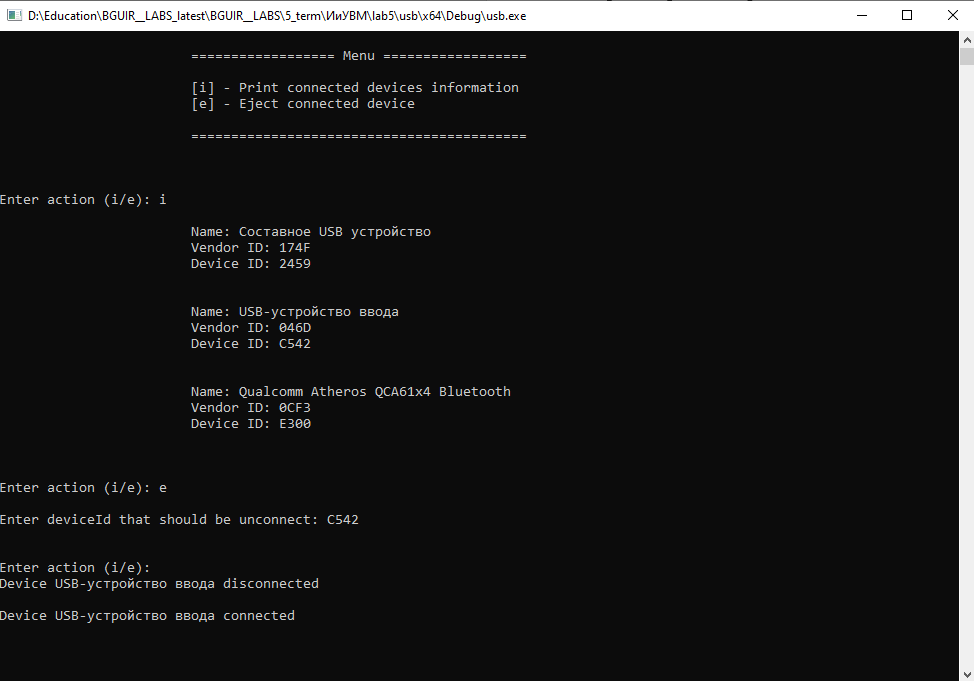


Рисунок 4.1 – Результат работы программы

1. **Вывод**

В ходе выполнения данной лабораторной работы были изучены теоретические сведения о USB-устройствах, их версиях, различных видах разъемов, способах увеличения разъемов. Была написана программа, которая выводит список всех подключенных USB-устройств, дает возможность безопасно извлечь устройство, а в случае его занятости сигнализирует об ошибке.

Программа написана для ОС Windows 10, использованные инструменты – язык программирования C++, Microsoft Visual Studio 2022.