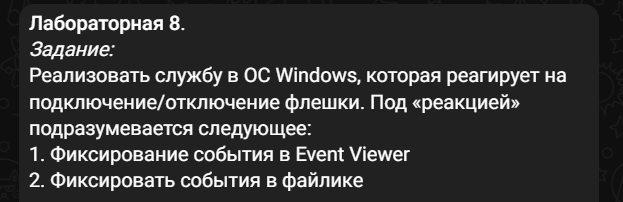
**Лабораторная работа 8**

**Студент:** Смирнов Константин ПКС-303

**Учитель:** Хусточка Алёна Витальевна



#include <Windows.h>

#include <tchar.h>

#include <stdio.h>

#include <time.h>

#include <string.h>

#include <tchar.h>

#include <strsafe.h>

#include <dbt.h>

#include <iostream>

#include <wtsapi32.h>

#include <Setupapi.h>

#include <devguid.h>

#define SERVICE\_NAME "Consta\_service"

SERVICE\_STATUS g\_ServiceStatus = { 0 };

SERVICE\_STATUS\_HANDLE g\_StatusHandle = NULL;

HANDLE g\_ServiceStopEvent = INVALID\_HANDLE\_VALUE;

HDEVNOTIFY hDeviceNotify;

GUID usbGUID = { 0x25dbce51, 0x6c8f, 0x4a72, 0x8a, 0x6d, 0xb5, 0x4c, 0x2b, 0x4f, 0xc8, 0x35 };

VOID WINAPI ServiceMain(DWORD argc, LPTSTR\* argv);

DWORD WINAPI ServiceCtrlHandler(DWORD dwOpcode, DWORD evtype, PVOID evdata, PVOID Context);

DWORD WINAPI ServiceWorkerThread(LPVOID lpParam);

//Запись в журнал событий

VOID SvcReportEvent(LPTSTR mes, LPTSTR mes1, LPTSTR mes2, LPTSTR mes3)

{

HANDLE hEventSource;

LPCTSTR lpszStrings[5];

hEventSource = RegisterEventSource(NULL, L"Consta\_service");

if (NULL != hEventSource)

{

lpszStrings[0] = L"From Consta\_service";

lpszStrings[1] = mes;

lpszStrings[2] = mes1;

lpszStrings[3] = mes2;

lpszStrings[4] = mes3;

ReportEvent(hEventSource, EVENTLOG\_INFORMATION\_TYPE, 0, 00, NULL, 5, 0, lpszStrings, NULL);

DeregisterEventSource(hEventSource);

}

}

//Установка службы

VOID SvcInstall()

{

SC\_HANDLE schSCManager;

SC\_HANDLE schService;

schSCManager = OpenSCManager(NULL, NULL, SC\_MANAGER\_ALL\_ACCESS);

if (NULL == schSCManager)

{

printf("OpenSCManager failed (%d)\n", GetLastError());

return;

}

schService = CreateService(schSCManager, L"Consta\_service", L"Consta\_service", SERVICE\_ALL\_ACCESS, SERVICE\_WIN32\_OWN\_PROCESS, SERVICE\_DEMAND\_START,

SERVICE\_ERROR\_NORMAL, L"C:\\VS\\PR8\_1\\x64\\Debug\\PR8\_1.exe", NULL, NULL, NULL, NULL, NULL);

if (schService == NULL)

{

printf("CreateService failed (%d)\n", GetLastError());

CloseServiceHandle(schSCManager);

return;

}

else printf("Service installed successfully\n");

CloseServiceHandle(schService);

CloseServiceHandle(schSCManager);

}

//Удаление службы

VOID \_\_stdcall DoDeleteSvc()

{

SC\_HANDLE schSCManager;

SC\_HANDLE schService;

SERVICE\_STATUS ssStatus;

schSCManager = OpenSCManager(NULL, NULL, SC\_MANAGER\_ALL\_ACCESS);

if (NULL == schSCManager)

{

printf("OpenSCManager failed (%d)\n", GetLastError());

return;

}

schService = OpenService(schSCManager, L"Consta\_service", DELETE);

if (schService == NULL)

{

printf("OpenService failed (%d)\n", GetLastError());

CloseServiceHandle(schSCManager);

return;

}

if (!DeleteService(schService))

{

printf("DeleteService failed (%d)\n", GetLastError());

}

else printf("Service deleted successfully\n");

CloseServiceHandle(schService);

CloseServiceHandle(schSCManager);

}

//Регистрация подключения USB

BOOL DoRegisterDeviceInterfaceToHwnd(IN GUID InterfaceClassGuid, OUT HDEVNOTIFY\* hDeviceNotify)

{

DEV\_BROADCAST\_DEVICEINTERFACE NotificationFilter;

ZeroMemory(&NotificationFilter, sizeof(NotificationFilter));

NotificationFilter.dbcc\_size = sizeof(DEV\_BROADCAST\_DEVICEINTERFACE);

NotificationFilter.dbcc\_devicetype = DBT\_DEVTYP\_DEVICEINTERFACE;

NotificationFilter.dbcc\_classguid = InterfaceClassGuid;

\*hDeviceNotify = RegisterDeviceNotification(g\_StatusHandle, &NotificationFilter, DEVICE\_NOTIFY\_SERVICE\_HANDLE | DEVICE\_NOTIFY\_ALL\_INTERFACE\_CLASSES);

if (NULL == \*hDeviceNotify)

{

return FALSE;

}

return TRUE;

}

int \_tmain(int argc, TCHAR\* argv[])

{

//Команды удаления и установки службы

if (lstrcmpW(argv[1], L"remove") == 0)

{

DoDeleteSvc();

}

if (lstrcmpW(argv[1], L"install") == 0)

{

SvcInstall();

}

SERVICE\_TABLE\_ENTRY ServiceTable[] =

{

{(LPWSTR)SERVICE\_NAME, (LPSERVICE\_MAIN\_FUNCTION)ServiceMain},

{NULL, NULL}

};

if (StartServiceCtrlDispatcher(ServiceTable) == FALSE)

{

return GetLastError();

}

return 0;

}

VOID WINAPI ServiceMain(DWORD argc, LPTSTR\* argv)

{

DWORD Status =

E\_FAIL;

g\_StatusHandle = RegisterServiceCtrlHandlerEx((LPCWSTR)SERVICE\_NAME, ServiceCtrlHandler, NULL);

DoRegisterDeviceInterfaceToHwnd(usbGUID, &hDeviceNotify);

if (g\_StatusHandle == NULL)

{

}

ZeroMemory(&g\_ServiceStatus, sizeof(g\_ServiceStatus));

g\_ServiceStatus.dwServiceType = SERVICE\_WIN32\_OWN\_PROCESS;

g\_ServiceStatus.dwControlsAccepted = 0;

g\_ServiceStatus.dwCurrentState = SERVICE\_START\_PENDING;

g\_ServiceStatus.dwWin32ExitCode = 0;

g\_ServiceStatus.dwServiceSpecificExitCode = 0;

g\_ServiceStatus.dwCheckPoint = 0;

if (SetServiceStatus(g\_StatusHandle, &g\_ServiceStatus) == FALSE)

{

}

g\_ServiceStopEvent = CreateEvent(NULL, TRUE, FALSE, NULL);

if (g\_ServiceStopEvent == NULL)

{

g\_ServiceStatus.dwControlsAccepted = 0;

g\_ServiceStatus.dwCurrentState = SERVICE\_STOPPED;

g\_ServiceStatus.dwWin32ExitCode = GetLastError();

g\_ServiceStatus.dwCheckPoint = 1;

if (SetServiceStatus(g\_StatusHandle, &g\_ServiceStatus) == FALSE)

{

}

}

g\_ServiceStatus.dwControlsAccepted = SERVICE\_ACCEPT\_STOP;

g\_ServiceStatus.dwCurrentState = SERVICE\_RUNNING;

g\_ServiceStatus.dwWin32ExitCode = 0;

g\_ServiceStatus.dwCheckPoint = 0;

if (SetServiceStatus(g\_StatusHandle, &g\_ServiceStatus) == FALSE)

{

}

HANDLE hThread = CreateThread(NULL, 0, ServiceWorkerThread, NULL, 0, NULL);

WaitForSingleObject(hThread, INFINITE);

CloseHandle(g\_ServiceStopEvent);

g\_ServiceStatus.dwControlsAccepted = 0;

g\_ServiceStatus.dwCurrentState = SERVICE\_STOPPED;

g\_ServiceStatus.dwWin32ExitCode = 0;

g\_ServiceStatus.dwCheckPoint = 3;

if (SetServiceStatus(g\_StatusHandle, &g\_ServiceStatus) == FALSE)

{

}

}

//Работа с USB

DWORD WINAPI ServiceCtrlHandler(DWORD dwOpcode, DWORD evtype, PVOID evdata, PVOID Context)

{

switch (dwOpcode)

{

case SERVICE\_CONTROL\_DEVICEEVENT: {

PDEV\_BROADCAST\_DEVICEINTERFACE b = (PDEV\_BROADCAST\_DEVICEINTERFACE)evdata;

TCHAR strBuff[256];

DWORD bytesWritten;

SP\_DEVINFO\_DATA DeviceInfoData;

switch (evtype)

{

case DBT\_DEVICEARRIVAL:

{

for (int i = 0; i < lstrlenW(b->dbcc\_name); i++)

{

if (b->dbcc\_name[i] == '#')

{

b->dbcc\_name[i] = '/';

}

}

WCHAR VID[10] = { 0 };

WCHAR PID[10] = { 0 };

WCHAR GUID[40] = { 0 };

WCHAR\* fullname = &b->dbcc\_name[4];

HANDLE file = CreateFile(L"C:\\Users\\Home\\Desktop\\Plusiki\\8\\log.txt", GENERIC\_WRITE, FILE\_SHARE\_READ, NULL, CREATE\_ALWAYS, NULL, NULL);

WriteFile(file, &b->dbcc\_name, lstrlenW(b->dbcc\_name) \* 2, &bytesWritten, NULL);

int tempGUIDpos = lstrlenW(fullname) - 37;

wcsncpy\_s(VID, &fullname[8], 4);

wcsncpy\_s(PID, &fullname[17], 4);

wcsncpy\_s(GUID, &fullname[tempGUIDpos], 36);

WCHAR mes[100] = { 0 };

WCHAR tempmes1[100] = { 0 };

WCHAR tempmes2[100] = { 0 };

WCHAR tempmes3[100] = { 0 };

lstrcatW(mes, L"This USB was connected: ");

lstrcatW(tempmes1, L"VID = ");

lstrcatW(tempmes1, VID);

lstrcatW(tempmes2, L"PID = ");

lstrcatW(tempmes2, PID);

lstrcatW(tempmes3, L"GUID = ");

lstrcatW(tempmes3, GUID);

SvcReportEvent(mes, tempmes1, tempmes2, tempmes3);

}

break;

case DBT\_DEVICEREMOVECOMPLETE:

{

for (int i = 0; i < lstrlenW(b->dbcc\_name); i++)

{

if (b->dbcc\_name[i] == '#')

{

b->dbcc\_name[i] = '/';

}

}

WCHAR VID[10] = { 0 };

WCHAR PID[10] = { 0 };

WCHAR GUID[40] = { 0 };

WCHAR\* fullname = &b->dbcc\_name[4];

int tempGUIDpos = lstrlenW(fullname) - 37;

wcsncpy\_s(VID, &fullname[8], 4);

wcsncpy\_s(PID, &fullname[17], 4);

wcsncpy\_s(GUID, &fullname[tempGUIDpos], 36);

WCHAR mes[100] = { 0 };

WCHAR tempmes1[100] = { 0 };

WCHAR tempmes2[100] = { 0 };

WCHAR tempmes3[100] = { 0 };

lstrcatW(mes, L"This USB was disconnected: ");

lstrcatW(tempmes1, L"VID = ");

lstrcatW(tempmes1, VID);

lstrcatW(tempmes2, L"PID = ");

lstrcatW(tempmes2, PID);

lstrcatW(tempmes3, L"GUID = ");

lstrcatW(tempmes3, GUID);

SvcReportEvent(mes, tempmes1, tempmes2, tempmes3);

}

break;

}

}

break;

case SERVICE\_CONTROL\_STOP:

if (g\_ServiceStatus.dwCurrentState != SERVICE\_RUNNING)

break;

g\_ServiceStatus.dwControlsAccepted = 0;

g\_ServiceStatus.dwCurrentState = SERVICE\_STOP\_PENDING;

g\_ServiceStatus.dwWin32ExitCode = 0;

g\_ServiceStatus.dwCheckPoint = 4;

if (SetServiceStatus(g\_StatusHandle, &g\_ServiceStatus) == FALSE)

{

}

SetEvent(g\_ServiceStopEvent);

break;

default:

break;

}

return 0;

}

DWORD WINAPI ServiceWorkerThread(LPVOID lpParam)

{

WaitForSingleObject(g\_ServiceStopEvent, INFINITE);

return ERROR\_SUCCESS;

}

