Практическая работа №7 Асылбек уулу Бакыта из группы ИСП-308

Служба

Постановка задачи:

Реализовать службу в ОС Windows, которая реагирует на подключение/отключение флешки. Под «реакцией» подразумевается следующее: фиксирование события в Event Viewer, фиксировать события в файлике

Примечание:

Нужно не только написать сообщение о том, что «флешка подключена/извлечена», но и указать в этом сообщении серийный номер флешки. Также сама служба должна уметь принимать и обрабатывать аргументы install и remove, чтобы была возможность установить и удалить службу из командной строки.

#include <windows.h>

#include <tchar.h>

#include <strsafe.h>

#include <dbt.h>

#pragma comment(lib, "advapi32.lib")

#define SVCNAME TEXT("SvcName")

SERVICE\_STATUS SvcStatus;

SERVICE\_STATUS\_HANDLE SvcStatusHandle;

HANDLE SvcStopEvent = NULL;

HDEVNOTIFY m\_hDevNotify = NULL;

VOID SvcInstall(void);

VOID WINAPI SvcCtrlHandler(DWORD dwOpcode, DWORD evtype, PVOID evdata, PVOID Context);

VOID WINAPI SvcMain(DWORD, LPTSTR\*);

VOID ReportSvcStatus(DWORD, DWORD, DWORD);

VOID SvcInit(DWORD, LPTSTR\*);

VOID SvcReportEvent(LPTSTR);

// Цель: Точка входа в процесс.

// Параметры: нет параметров.

// Возвращает значение: не возвращает значение, по умолчанию 0.

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

{

// Если параметр командной строки "установить", установите службу.

// В противном случае служба, возможно, запускается SCM.

if (lstrcmpi(argv[1], TEXT("install")) == 0)

{

SvcInstall();

return 0;

}

// TO\_DO: Добавить в эту таблицу любые дополнительные службы для процесса.

SERVICE\_TABLE\_ENTRY DispatchTable[] = {{ (LPWSTR)SVCNAME, (LPSERVICE\_MAIN\_FUNCTION)SvcMain }, { NULL, NULL }};

// Этот вызов возвращается, когда служба остановлена. Процесс должен просто завершиться, когда вызов вернется.

if (!StartServiceCtrlDispatcher(DispatchTable))

{

SvcReportEvent((LPTSTR)TEXT("StartServiceCtrlDispatcher"));

}

}

// Цель: Установить службу в базе данных SCM.

// Параметры: нет параметров.

// Возвращает значение: не возвращает значение.

VOID SvcInstall()

{

SC\_HANDLE SCManager;

SC\_HANDLE Service;

TCHAR UnquotedPath[MAX\_PATH];

if (!GetModuleFileName(NULL, UnquotedPath, MAX\_PATH))

{

printf("Can't install service\n");

return;

}

// В случае, если путь содержит пробел, он должен быть заключен в кавычки, чтобы он был правильно интерпретирован.

// Например, "d:\my share\myservice.exe" должно быть указано как ""d:\my share\myservice.exe"".

TCHAR Path[MAX\_PATH];

StringCbPrintf(Path, MAX\_PATH, TEXT("\"%s\""), UnquotedPath);

// Получить доступ к базе данных SCM.

SCManager = OpenSCManager(NULL, // Имя компьютера.

NULL, // Имя базы данных диспетчера управления службами.

SC\_MANAGER\_ALL\_ACCESS); // Полный доступ к диспетчеру управления службами.

if (NULL == SCManager)

{

printf("OpenSCManager failed\n");

return;

}

// Создать службу.

Service = CreateService(SCManager, // Дескриптор базы данных SCM.

SVCNAME, // Имя устанавливаемой службы.

SVCNAME, // Отображаемое имя, используемое программами UI для идентификации службы.

SERVICE\_ALL\_ACCESS, // Доступ к службе.

SERVICE\_WIN32\_OWN\_PROCESS, // Тип службы (Служба, которая выполняется в собственном процессе).

SERVICE\_DEMAND\_START, // Параметры запуска службы (Служба, запускаемая SCM, когда процесс вызывает функцию StartService.).

SERVICE\_ERROR\_NORMAL, // Серьезность ошибки и действия, предпринятые в случае сбоя запуска этой службы (Запускаемая программа регистрирует ошибку в журнале событий, но продолжает операцию запуска).

Path, // Полный путь к двоичному файлу службы.

NULL, // Служба не принадлежит к группе.

NULL, // Тег не изменяется.

NULL, // Служба не имеет зависимостей.

NULL, // CreateService использует учетную запись LocalSystem.

NULL); // Имя учетной записи, является именем управляемой учетной записи службы или именем виртуальной учетной записи.

if (Service == NULL)

{

printf("CreateService failed\n");

CloseServiceHandle(SCManager);

return;

}

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

CloseServiceHandle(Service);

CloseServiceHandle(SCManager);

}

// Цель: Точка входа в службу.

// Параметры: dwArgc - количество аргументов в массиве lpszArgv. lpszArgv - Массив строк.

// Первая строка - это имя службы, а последующие строки передаются процессом, который вызвал функцию startService для запуска службы.

// Возвращает значение: не возвращает значение.

VOID WINAPI SvcMain(DWORD dwArgc, LPTSTR\* lpszArgv)

{

// Зарегистрируйте функцию обработчика для службы

SvcStatusHandle = RegisterServiceCtrlHandlerEx(SVCNAME, (LPHANDLER\_FUNCTION\_EX)SvcCtrlHandler, 0);

if (!SvcStatusHandle)

{

SvcReportEvent((LPTSTR)TEXT("RegisterServiceCtrlHandler"));

return;

}

DEV\_BROADCAST\_DEVICEINTERFACE NotificationFilter;

ZeroMemory(&NotificationFilter, sizeof(NotificationFilter));

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

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

m\_hDevNotify = RegisterDeviceNotification(SvcStatusHandle, &NotificationFilter, DEVICE\_NOTIFY\_SERVICE\_HANDLE | DEVICE\_NOTIFY\_ALL\_INTERFACE\_CLASSES);

// Эти члены SERVICE\_STATUS остаются заданными здесь.

SvcStatus.dwServiceType = SERVICE\_WIN32\_OWN\_PROCESS;

SvcStatus.dwServiceSpecificExitCode = 0;

// Сообщить о первоначальном статусе в SCM.

ReportSvcStatus(SERVICE\_START\_PENDING, NO\_ERROR, 3000);

// Выполните инициализацию и работу, специфичную для конкретной службы.

SvcInit(dwArgc, lpszArgv);

}

// Цель: код службы

// Параметры: dwArgc - количество аргументов в массиве lpszArgv. lpszArgv - массив строк.

// Первая строка - это имя службы, а последующие строки передаются процессом, который вызвал функцию startService для запуска службы.

// Возвращает значение: не возвращает значение.

VOID SvcInit(DWORD dwArgc, LPTSTR\* lpszArgv)

{

// TO\_DO: Объявляем и устанавливаем любые необходимые переменные.

// Обязательно периодически вызывайте ReportSvcStatus() с SERVICE\_START\_PENDING.

// Если инициализация завершается неудачей, вызовите ReportSvcStatus с SERVICE\_STOPPED.

// Создать событие. Функция обработчика управления, SvcCtrlHandler, сигнализирует об этом событии, когда получает код управления остановкой.

SvcStopEvent = CreateEvent(NULL, // дескриптор безопасности по умолчанию.

TRUE, // функция создает объект события сброса вручную.

FALSE, // не сигнализируется.

NULL); // объект события создается без имени.

if (SvcStopEvent == NULL)

{

ReportSvcStatus(SERVICE\_STOPPED, GetLastError(), 0);

return;

}

// Сообщить о состоянии выполнения после завершения инициализации.

ReportSvcStatus(SERVICE\_RUNNING, NO\_ERROR, 0);

// TO\_DO: Выполнять работу до тех пор, пока служба не остановится.

while (1)

{

// Проверить, следует ли останавливать службу.

WaitForSingleObject(SvcStopEvent, INFINITE);

ReportSvcStatus(SERVICE\_STOPPED, NO\_ERROR, 0);

return;

}

}

// Цель: Установить текущее состояние службы и сообщает о нем SCM.

// Параметры: dwCurrentState - Текущее состояние (см. SERVICE\_STATUS), dwWin32ExitCode - Код системной ошибки,

// dwWaitHint - расчетное время ожидания операции, в миллисекундах.

// Возвращает значение: не возвращает значение.

VOID ReportSvcStatus(DWORD dwCurrentState, DWORD dwWin32ExitCode, DWORD dwWaitHint)

{

static DWORD CheckPoint = 1;

// Заполните структуру SERVICE\_STATUS.

SvcStatus.dwCurrentState = dwCurrentState;

SvcStatus.dwWin32ExitCode = dwWin32ExitCode;

SvcStatus.dwWaitHint = dwWaitHint;

if (dwCurrentState == SERVICE\_START\_PENDING)

SvcStatus.dwControlsAccepted = 0;

else SvcStatus.dwControlsAccepted = SERVICE\_ACCEPT\_STOP;

if ((dwCurrentState == SERVICE\_RUNNING) || (dwCurrentState == SERVICE\_STOPPED))

SvcStatus.dwCheckPoint = 0;

else SvcStatus.dwCheckPoint = CheckPoint++;

// Сообщите о статусе службы в SCM.

SetServiceStatus(SvcStatusHandle, &SvcStatus);

}

// Цль: Вызывается SCM всякий раз, когда управляющий код отправляется в службу с использованием функции ControlService.

// Параметры: dwCtrl - управляющий код.

// Возвращает значение: не возвращает значение.

VOID WINAPI SvcCtrlHandler(DWORD dwOpcode, DWORD evtype, PVOID evdata, PVOID Context)

{

HANDLE event\_log = RegisterEventSourceW(NULL, (LPCWSTR)"SvcName");

LPWSTR tab = (LPWSTR)"SvcNameID";

// Обработать запрошенный управляющий код.

switch (dwOpcode)

{

case SERVICE\_CONTROL\_STOP:

ReportSvcStatus(SERVICE\_STOP\_PENDING, NO\_ERROR, 0);

UnregisterDeviceNotification(m\_hDevNotify);

// Подать сигнал службе об остановке.

SetEvent(SvcStopEvent);

ReportSvcStatus(SvcStatus.dwCurrentState, NO\_ERROR, 0);

return;

case SERVICE\_CONTROL\_INTERROGATE:

break;

case SERVICE\_CONTROL\_DEVICEEVENT:

OutputDebugString((LPCWSTR)"SERVICE\_CONTROL\_DEVICEEVENT");

switch (evtype)

{

case DBT\_DEVICEREMOVECOMPLETE:

{

OutputDebugString((LPCWSTR)"The device is disconnected");

LPCWSTR message = L"The device is disconnected";

ReportEventW(event\_log, EVENTLOG\_SUCCESS, 0, 0, NULL, 1, 0, &message, NULL);

}

break;

case DBT\_DEVICEARRIVAL:

{

OutputDebugString((LPCWSTR)"The device is connected");

LPCWSTR message = L"The device is connected";

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

// Запись серийного номера в файл

HANDLE f = CreateFileW(L"C:\\text.TXT", // Имя создаваемого или открываемого файла

GENERIC\_ALL, // Запрашиваемый доступ к файлу.

0, // Предотвращает последующие операции открытия файла, если они запрашивают доступ к удалению чтению или записи.

NULL, // Дескриптор безопасности по умолчанию.

OPEN\_ALWAYS, // Всегда открывает файл.

0, // Атрибуты по умолчанию

0); // Нет шаблона дескриптора файла

DWORD Write = 0;

WriteFile(f, // Дескриптор файла

a->dbcc\_name, // Указатель на буфер, содержащий данные для записи в файл.

wcslen(a->dbcc\_name), // Количество байтов, записываемых в файл.

&Write, // Указатель на переменную, которая получает количество байтов.

NULL); // не перекрывающийся

ReportEvent(event\_log, EVENTLOG\_SUCCESS, 0, 0, NULL, 1, 0, &message, NULL);

CloseHandle(f);

}

break;

}

break;

default:

break;

}

}

// Цель: Записать сообщения в журнал событий.

// Параметры: szFunction - имя функции, которая завершилась с ошибкой.

// Возвращает значение: не возвращает значение.

// Примечания: Служба должна иметь запись в журнале событий приложения.

VOID SvcReportEvent(LPTSTR szFunction)

{

HANDLE EventSource;

LPCTSTR Strings[2];

TCHAR Buffer[80];

EventSource = RegisterEventSource(NULL, SVCNAME);

if (NULL != EventSource)

{

StringCchPrintf(Buffer, 80, TEXT("%s failed"), szFunction);

Strings[0] = SVCNAME;

Strings[1] = Buffer;

ReportEventW(EventSource, // Дескриптор журнала событий

EVENTLOG\_ERROR\_TYPE, // Тип регистрируемого события (Событие ошибки).

0, // Категория события.

1, // Идентификатор события.

NULL, // Идентификатор безопасности не требуется.

2, // Количество строк вставки в массиве.

0, // Количество байтов необработанных (двоичных) данных для записи в журнал.

Strings, // Указатель на буфер, содержащий массив строк с нулевым завершением.

NULL); // Указатель на буфер, содержащий двоичные данные.

DeregisterEventSource(EventSource);

}

}

