ЛАБОРАТОРНАЯ РАБОТА. Создание и регистрация сервиса.

**Цель**. Создать собственный сервис и запустить его.

**КРАТКИЕ СВЕДЕНИЯ**.

Сервисы (службы) – фоновые процессы в оперативной памяти. Их количество достаточно велико, так что они реально могут тормозить работу компьютера. Мы попытаемся создать свой сервис. Это будет консольное приложение определенной структуры. В нем должна быть главная функция, функция для обработки старта сервиса и его останова. Создаем консольное приложение для сервиса, но НЕ ВЫПОЛНЯЕМ ЕГО! Только компилируем. Вот текст этого приложения.

#include "stdafx.h"

#include "windows.h"

SERVICE\_STATUS serviceStatus;

SERVICE\_STATUS\_HANDLE serviceStatusHandle;

DWORD ServiceInitialization(DWORD argc, LPTSTR \*argv, DWORD \*specificError);

// Application define ServiceMain()...

void WINAPI ServiceStart(DWORD argc, LPTSTR \*argv);

void WINAPI ServiceCtrlHandler(DWORD opcode);

void SvcDebugOut(LPSTR String, DWORD Status);

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

{

SERVICE\_TABLE\_ENTRY ServiceTable[1];

ServiceTable[0].lpServiceName = TEXT("SERV\_G");

ServiceTable[0].lpServiceProc = (LPSERVICE\_MAIN\_FUNCTION)ServiceStart;

if(!StartServiceCtrlDispatcher(ServiceTable))

{

printf("Error in registration:%x",GetLastError());

}

else

{

printf("Registration OK");

}

getchar();

return 0;

}

DWORD ServiceInitialization(DWORD argc, LPTSTR \*argv, DWORD \*specificError)

{

\*argv;

argc;

specificError;

// should return status...

return 0;

}

void WINAPI ServiceStart(DWORD argc, LPTSTR \*argv)

{

DWORD status;

DWORD specificError;

// Type of service, application or driver...

serviceStatus.dwServiceType = SERVICE\_WIN32;

// The service is starting...

serviceStatus.dwCurrentState = SERVICE\_START\_PENDING;

// The service can be stopped & can be paused and continued.

serviceStatus.dwControlsAccepted = SERVICE\_ACCEPT\_STOP | SERVICE\_ACCEPT\_PAUSE\_CONTINUE;

serviceStatus.dwWin32ExitCode = 0;

serviceStatus.dwServiceSpecificExitCode = 0;

serviceStatus.dwCheckPoint = 0;

serviceStatus.dwWaitHint = 0;

serviceStatusHandle = RegisterServiceCtrlHandler(TEXT("SERV\_G"), ServiceCtrlHandler);

if (serviceStatusHandle == (SERVICE\_STATUS\_HANDLE)0)

{

SvcDebugOut("[MY\_SERVICE] RegisterServiceCtrlHandler() failed, error: %d.\n", GetLastError());

return;

}

else

printf("RegisterServiceCtrlHandler() looks OK.\n");

//===============================================================

// Initialization code goes here. Return the status...

status = ServiceInitialization(argc, argv, &specificError);

// Handle error condition

if (status != NO\_ERROR)

{

// The service is not running...

serviceStatus.dwCurrentState = SERVICE\_STOPPED;

serviceStatus.dwCheckPoint = 0;

serviceStatus.dwWaitHint = 0;

serviceStatus.dwWin32ExitCode = status;

serviceStatus.dwServiceSpecificExitCode = specificError;

SetServiceStatus(serviceStatusHandle, &serviceStatus);

return;

}

// Initialization complete - report running status.

serviceStatus.dwCurrentState = SERVICE\_RUNNING;

serviceStatus.dwCheckPoint = 0;

serviceStatus.dwWaitHint = 0;

if (!SetServiceStatus(serviceStatusHandle, &serviceStatus))

{

status = GetLastError();

SvcDebugOut("SetServiceStatus() error: %ld\n", status);

}

else

printf("SetServiceStatus() looks OK.\n");

// By assuming starting and initializing the service is OK, this

// is where the service does its work...

SvcDebugOut("Returning the Main Thread.\n", 0);

return;

}

void WINAPI ServiceCtrlHandler(DWORD opcode)

{

// Just a skeleton here...

}

void SvcDebugOut(LPSTR String, DWORD Status)

{

CHAR Buffer[1024];

if (strlen(String) < 1000)

{

sprintf(Buffer, String, Status);

OutputDebugStringA(Buffer);

}

else

printf("String too long...\n");

}

Сервис регистрируется при запуске StartServiceCtrlDispatcher. Наша задача построить exe файл. Основная процедура сервиса это ServiceStart.

Логика этой функции проста. Сначала регистрируем функцию которая будет обрабатывать управляющие запросы от SCM (диспетчера сервисов), например, запрос на остановку. Регистрация производиться при помощи функции **RegisterServiceCtrlHandler.** Для изменения статуса сервиса используется функция **SetServiceStatus**.

Сам сервис называется SERV\_G. Все это указывается в строках

SERVICE\_TABLE\_ENTRY ServiceTable[1];

ServiceTable[0].lpServiceName = TEXT("SERV\_G");

ServiceTable[0].lpServiceProc = (LPSERVICE\_MAIN\_FUNCTION)ServiceStart;

if(!StartServiceCtrlDispatcher(ServiceTable))

Теперь опишем функцию по обработке запросов (в примере выше эта функция пустая). Можно предложить такой вариант:

void ServiceCtrlHandler (DWORD request) {   
  switch(request)   
  {   
    case SERVICE\_CONTROL\_STOP:   
      addLogMessage("Stopped.");  
  
      serviceStatus.dwWin32ExitCode = 0;   
      serviceStatus.dwCurrentState = SERVICE\_STOPPED;   
      SetServiceStatus (serviceStatusHandle, &serviceStatus);  
      return;   
  
    case SERVICE\_CONTROL\_SHUTDOWN:   
      addLogMessage("Shutdown.");  
  
      serviceStatus.dwWin32ExitCode = 0;   
      serviceStatus.dwCurrentState = SERVICE\_STOPPED;   
      SetServiceStatus (serviceStatusHandle, &serviceStatus);  
      return;   
      
    default:  
      break;  
  }   
  
  SetServiceStatus (serviceStatusHandle, &serviceStatus);  
  
  return;   
}

Особенно здесь ничего и нет. Вставлена обработка событий завершения сервиса.

**Установка сервиса**

Есть несколько вариантов, один из них, при помощи утилита **sc (скачайте)**. Установка производиться следующей командой:

sc create SERV\_G binpath= c:\SERV\_G.exe

**УДАЛЕНИЕ СЕРВИСА**

sc delete SampleService

**ЗАДАНИЕ.**

Выполните то, что есть в этой лабораторной. Попробуйте зарегистрировать и запустить сервис. Запуск выполните из окна служб. ВНИМАНИЕ: запрограммируйте сервис так, чтобы при его останове выполнялось завершение работы системы