Лабораторная работа 3

СЕРВИСЫ MS WINDOWS

Цель работы:

Получить навыки работы с сервисами Windows

Задача:

Написать программу, работающую в качестве сервиса (службы) Windows (версии 7-10). Программа должна осуществлять резервное копирование данных из указанной директории.

Требования:

- Программа не должна быть интерактивной, все настройки задаются с помощью конфигурационного файла;

- Программа должна создавать архив с резервной копией данных из указанной директории;

- Файлы, включаемые в резервную копию, задаются по маске (с использованием символов *, ?) или непосредственно по именам;

- Резервная копия должна создаваться в виде архива формата ZIP;

- При изменении существующих или появлении новых файлов в исходной директории (соответствующих заданной маске) архив с резервной копией автоматически дополняется новыми данными (или заменяются существующие);

- Инсталляция, деинсталляция, запуск, останов из командной строки (реализация своего SCP);

- Корректный останов и перезапуск службы по запросу от SCM (обработка команд, передаваемых через оснастку "Службы");

- Имена файлов и директорий задаются в формате UNC.

Конфигурационный файл как минимум должен содержать:

- Имя директории, из которой выполняется резервное копирование;

- Имя файла-архива и директории, в которой он должен находиться;

- Маски/имена файлов, помещаемых в резервную копию.

Теоретические сведения

Сервис – это системный процесс, который предоставляет функциональность, не увязываемую с интерактивным пользователем. Windows-сервисы состоят из трех компонентов:

Сервисное приложение – это программа с дополнительным кодом для обработки команд от SCM и возврата ему статусной информации.

Программа управления сервисом (SCP) – это приложение, использующее SCM функции управления сервисами CreateService, OpenService, StartService, DeleteService.

Service Control Manager (SCM) (\Windows\System32\Services.exe) — в Microsoft Windows особый системный процесс, реализующий технологию удалённого вызова процедур (remote procedure call — RPC). Обеспечивает создание, удаление, запуск и остановку служб ОС. Запускается процессом winlogon при старте системы.

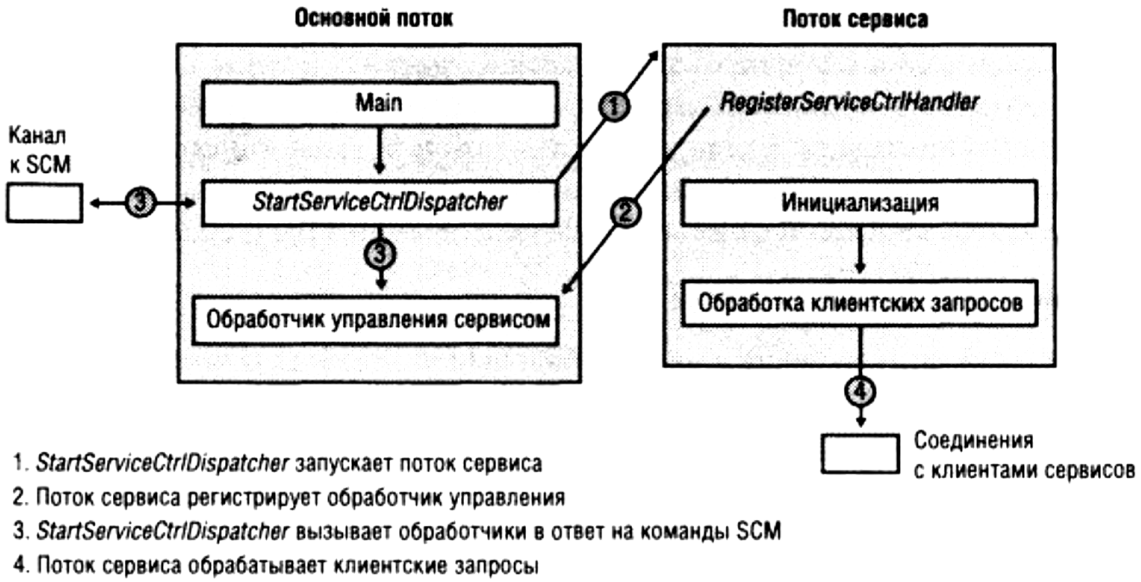
SCM отвечает за:

- Последнюю удачную конфигурацию;
- Создание, удаление, назначение букв сетевым дискам;
- Управление сервисами.

Когда вы устанавливаете приложение, в состав которого входит сервис, программа установки приложения должна зарегистрировать этот сервис в системе. Для его регистрации вызывается Windows-функция Create-Service, реализованная в Advapi32.dll (\Windows\System32\Advapi32.dll). Эта DLL реализует всю клиентскую часть SCM API. Регистрируя сервис через CreateService, программа установки посылает SCM сообщение о том, где будет находиться данный сервис. Затем SCM создает в реестре раздел для сервиса по адресу HKLM\SYSTEM\CurrentControl-Set\Services. Зарегистрировав сервис, программа установки или управляющее приложение может запустить его через функцию StartService. При вызове CreateService программа должна указывать некоторые параметры, описывающие характеристики сервиса. Эти характеристики включают тип сервиса, местонахождение его исполняемого файла, необязательное экранное имя, необязательные имя и пароль для запуска сервиса в контексте защиты определенной учетной записи, тип запуска, код, указывающий, как система должна реагировать на ошибку при запуске сервиса, и необязательную информацию о моменте запуска относительно других сервисов. SCM хранит каждую характеристику как параметр в разделе, созданном для данного сервиса.

SCM посылает следующие команды:

1. Stop (остановка)
2. Pause (пауза)
3. Resume (возобновление)
4. Interrogate (опрос)
5. Shutdown (выключение)
6. Команды, определенные приложением



Руководство к реализации:

Сервис начинается с функции _tmain()

```
int _tmain(int argc, _TCHAR* argv[]) {
  SERVICE_TABLE_ENTRY ServiceTable[1];
  ServiceTable[0].lpServiceName = serviceName;
  ServiceTable[0].lpServiceProc = (LPSERVICE_MAIN_FUNCTION)ServiceMain;

  StartServiceCtrlDispatcher(ServiceTable);
}
```

SERVICE_TABLE_ENTRY это структура, которая описывает точку входа для сервис менеджера, в данном случае вход будет происходить через функцию ServiceMain. Функция StartServiceCtrlDispatcher связывает сервис с SCM

Точка входа в сервис:

Прежде, чем описывать функцию, нам понадобятся две глобальные переменные:

```
SERVICE_STATUS ServiceStatus;
SERVICE_STATUS_HANDLE hStatus;
```

Структура SERVICE_STATUS используется для оповещения SCM о текущем статусе сервиса. Ниже приведен код функции ServiceMain:

```
void ServiceMain(int argc, char** argv) {
  int error;
  int i = 0;

  serviceStatus.dwServiceType    = SERVICE_WIN32_OWN_PROCESS;
  serviceStatus.dwCurrentState    = SERVICE_START_PENDING;
```

```
  serviceStatus.dwControlsAccepted  = SERVICE_ACCEPT_STOP |
SERVICE_ACCEPT_SHUTDOWN;
  serviceStatus.dwWin32ExitCode   = 0;
  serviceStatus.dwServiceSpecificExitCode = 0;
  serviceStatus.dwCheckPoint     = 0;
  serviceStatus.dwWaitHint      = 0;

  serviceStatusHandle = RegisterServiceCtrlHandler(serviceName,
(LPHANDLER_FUNCTION)ControlHandler);
  if (serviceStatusHandle == (SERVICE_STATUS_HANDLE)0) {
    return;
  }

  error = InitService();
  if (error) {
    serviceStatus.dwCurrentState    = SERVICE_STOPPED;
    serviceStatus.dwWin32ExitCode   = -1;
    SetServiceStatus(serviceStatusHandle, &serviceStatus);
    return;
  }

  serviceStatus.dwCurrentState = SERVICE_RUNNING;
  SetServiceStatus (serviceStatusHandle, &serviceStatus);

  while (serviceStatus.dwCurrentState == SERVICE_RUNNING)
  {
    char buffer[255];
    sprintf_s(buffer, "%u", i);
    int result = addLogMessage(buffer);
    if (result)  {
      serviceStatus.dwCurrentState    = SERVICE_STOPPED;
      serviceStatus.dwWin32ExitCode   = -1;
      SetServiceStatus(serviceStatusHandle, &serviceStatus);
      return;
    }
    i++;
  }

  return;
}
```

Логика этой функции проста. Сначала регистрируется функция, которая будет обрабатывать управляющие запросы от SCM, например, запрос на остановку. Регистрация производиться при помощи функции RegisterServiceCtrlHandler. При корректном запуске сервиса пишем в файл значения переменой i.
Для изменения статуса сервиса используется функция SetServiceStatus.

Примечание: функция addLogMessage:

```
int addLogMessage(char* str, int code) {
        errno_t err;
        FILE* log;
```

```
	if ((err = fopen_s(&log, /*Filename*/, "a+")) != 0) {
		return -1;
	}

	fprintf(log, "[code: %u] %s\n", code, str);
	fclose(log);
	return 0;
}
```

Теперь опишем функцию по обработке запросов:

```
void ControlHandler(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;
}
```

ControlHandler вызывается каждый раз, как SCM шлет запросы на изменение состояния сервиса. В основном ее используют для описания корректного завершения работы сервиса.

Установка сервиса:

Изменим немного логику функции _tmain:

```
int _tmain(int argc, _TCHAR* argv[]) {

  servicePath = LPTSTR(argv[0]);

  if(argc - 1 == 0) {
    SERVICE_TABLE_ENTRY ServiceTable[1];
    ServiceTable[0].lpServiceName = serviceName;
    ServiceTable[0].lpServiceProc = (LPSERVICE_MAIN_FUNCTION)ServiceMain;
```

```
    if(!StartServiceCtrlDispatcher(ServiceTable)) {
      addLogMessage("Error: StartServiceCtrlDispatcher");
    }
  } else if( wcscmp(argv[argc-1], _T("install")) == 0) {
    InstallService();
  } else if( wcscmp(argv[argc-1], _T("remove")) == 0) {
    RemoveService();
  } else if( wcscmp(argv[argc-1], _T("start")) == 0 ){
    StartService();
  }
}
```

Теперь у нас появятся еще 3 функции:

```
int InstallService() {
  SC_HANDLE hSCManager = OpenSCManager(NULL, NULL, SC_MANAGER_CREATE_SERVICE);
  if(!hSCManager) {
    addLogMessage("Error: Can't open Service Control Manager");
    return -1;
  }

  SC_HANDLE hService = CreateService(
    hSCManager,
    serviceName,
    serviceName,
    SERVICE_ALL_ACCESS,
    SERVICE_WIN32_OWN_PROCESS,
    SERVICE_DEMAND_START,
    SERVICE_ERROR_NORMAL,
    servicePath,
    NULL, NULL, NULL, NULL, NULL
  );

  if(!hService) {
    int err = GetLastError();
    switch(err) {
      case ERROR_ACCESS_DENIED:
        addLogMessage("Error: ERROR_ACCESS_DENIED");
        break;
      case ERROR_CIRCULAR_DEPENDENCY:
        addLogMessage("Error: ERROR_CIRCULAR_DEPENDENCY");
        break;
      case ERROR_DUPLICATE_SERVICE_NAME:
        addLogMessage("Error: ERROR_DUPLICATE_SERVICE_NAME");
        break;
      case ERROR_INVALID_HANDLE:
        addLogMessage("Error: ERROR_INVALID_HANDLE");
        break;
      case ERROR_INVALID_NAME:
        addLogMessage("Error: ERROR_INVALID_NAME");
        break;
      case ERROR_INVALID_PARAMETER:
        addLogMessage("Error: ERROR_INVALID_PARAMETER");
        break;
      case ERROR_INVALID_SERVICE_ACCOUNT:
        addLogMessage("Error: ERROR_INVALID_SERVICE_ACCOUNT");
        break;
      case ERROR_SERVICE_EXISTS:
        addLogMessage("Error: ERROR_SERVICE_EXISTS");
```

```
      break;
    default:
      addLogMessage("Error: Undefined");
   }
   CloseServiceHandle(hSCManager);
   return -1;
  }
  CloseServiceHandle(hService);

  CloseServiceHandle(hSCManager);
  addLogMessage("Success install service!");
  return 0;
}

int RemoveService() {
  SC_HANDLE hSCManager = OpenSCManager(NULL, NULL, SC_MANAGER_ALL_ACCESS);
  if(!hSCManager) {
    addLogMessage("Error: Can't open Service Control Manager");
    return -1;
  }
  SC_HANDLE hService = OpenService(hSCManager, serviceName, SERVICE_STOP | DELETE);
  if(!hService) {
    addLogMessage("Error: Can't remove service");
    CloseServiceHandle(hSCManager);
    return -1;
  }

  DeleteService(hService);
  CloseServiceHandle(hService);
  CloseServiceHandle(hSCManager);
  addLogMessage("Success remove service!");
  return 0;
}

int StartService() {
  SC_HANDLE hSCManager = OpenSCManager(NULL, NULL, SC_MANAGER_CREATE_SERVICE);
  SC_HANDLE hService = OpenService(hSCManager, serviceName, SERVICE_START);
  if(!StartService(hService, 0, NULL)) {
    CloseServiceHandle(hSCManager);
    addLogMessage("Error: Can't start service");
    return -1;
  }

  CloseServiceHandle(hService);
  CloseServiceHandle(hSCManager);
  return 0;
}
```

Теперь мы можем устанавливать, удалять и запускать сервис, не прибегая к различным утилитам:

```
SampleService.exe install
SampleService.exe remove
SampleService.exe start
```

Контрольные вопросы:

- Какие операции выполняются при установке сервиса в систему?

- Что такое SCP и зачем необходимо его реализовывать?

- Какие возможности по управлению сервисами НЕ предоставляет оснастка "Службы"?