这一节学习编写服务管理工具程序

# 学习大纲

|  |
| --- |
|  |

## OpenSCManager函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 与指定计算机上的服务控制管理器建立连接，并打开指定的服务控制管理器数据库。 语法 C++  SC\_HANDLE OpenSCManagerA(  [in, optional] LPCSTR lpMachineName,  [in, optional] LPCSTR lpDatabaseName,  [in] DWORD dwDesiredAccess  ); 参数 [in, optional] lpMachineName  目标计算机的名称。 如果指针为 NULL 或指向空字符串，则函数将连接到本地计算机上的服务控制管理器。  [in, optional] lpDatabaseName  服务控制管理器数据库的名称。 此参数应设置为 SERVICES\_ACTIVE\_DATABASE。 如果为 NULL，则默认打开SERVICES\_ACTIVE\_DATABASE数据库。  [in] dwDesiredAccess  对服务控制管理器的访问。 有关访问权限的列表，请参阅 [服务安全性和访问权限](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-security-and-access-rights)。  在授予请求的访问权限之前，系统会根据与服务控制管理器关联的安全描述符的任意访问控制列表检查调用进程的访问令牌。  SC\_MANAGER\_CONNECT访问权限是通过调用此函数隐式指定的。 返回值 如果函数成功，则返回值是指定服务控制管理器数据库的句柄。  如果函数失败，则返回值为 NULL。 要获得更多的错误信息，请调用 GetLastError。  SCM 可以设置以下错误代码。 其他错误代码可由 SCM 调用的注册表函数设置。  展开表   |  |  | | --- | --- | | **返回代码** | **说明** | | **ERROR\_ACCESS\_DENIED** | 请求的访问被拒绝。 | | **ERROR\_DATABASE\_DOES\_NOT\_EXIST** | 指定的数据库不存在。 |  注解 当进程使用 **OpenSCManager** 函数打开服务控制管理器数据库的句柄时，系统会在授予请求的访问权限之前执行安全检查。 有关详细信息，请参阅 [服务安全性和访问权限](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-security-and-access-rights)。  如果当前用户在连接到另一台计算机上的服务时没有适当的访问权限， **则 OpenSCManager** 函数调用将失败。 若要远程连接到服务，请使用 LOGON32\_LOGON\_NEW\_CREDENTIALS 调用 [LogonUser](https://learn.microsoft.com/zh-cn/windows/desktop/api/winbase/nf-winbase-logonusera) 函数，然后在调用 **OpenSCManager** 之前调用 [ImpersonateLoggedOnUser](https://learn.microsoft.com/zh-cn/windows/desktop/api/securitybaseapi/nf-securitybaseapi-impersonateloggedonuser)。 有关远程连接到服务的详细信息，请参阅 [服务和 RPC/TCP](https://learn.microsoft.com/zh-cn/windows/desktop/Services/services-and-rpc-tcp)。  只有具有管理员权限的进程才能打开可由 [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea) 函数使用的数据库句柄。  返回的句柄仅对调用 **OpenSCManager** 函数的进程有效。 可以通过调用 [CloseServiceHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-closeservicehandle) 函数将其关闭。 示例 有关示例，请参阅 [更改服务的配置](https://learn.microsoft.com/zh-cn/windows/desktop/Services/changing-a-service-configuration)。  **备注**  winsvc.h 标头将 OpenSCManager 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名的使用与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 要求  |  |  | | --- | --- | | **标头** | winsvc.h (包括 Windows.h) | | **Library** | Advapi32.lib | | **DLL** | Advapi32.dll |  另请参阅 [CloseServiceHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-closeservicehandle)  [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea)  [EnumServicesStatusEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-enumservicesstatusexa)  [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea)  [SCM 句柄](https://learn.microsoft.com/zh-cn/windows/desktop/Services/scm-handles)  [服务函数](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-functions) |

## CreateService函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 创建服务对象并将其添加到指定的服务控制管理器数据库。 语法 C++复制  SC\_HANDLE CreateServiceA(  [in] SC\_HANDLE hSCManager,  [in] LPCSTR lpServiceName,  [in, optional] LPCSTR lpDisplayName,  [in] DWORD dwDesiredAccess,  [in] DWORD dwServiceType,  [in] DWORD dwStartType,  [in] DWORD dwErrorControl,  [in, optional] LPCSTR lpBinaryPathName,  [in, optional] LPCSTR lpLoadOrderGroup,  [out, optional] LPDWORD lpdwTagId,  [in, optional] LPCSTR lpDependencies,  [in, optional] LPCSTR lpServiceStartName,  [in, optional] LPCSTR lpPassword  ); 参数 [in] hSCManager  服务控制管理器数据库的句柄。 此句柄由 [OpenSCManager](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openscmanagera) 函数返回，必须具有 **SC\_MANAGER\_CREATE\_SERVICE** 访问权限。 有关详细信息，请参阅 [服务安全性和访问权限](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-security-and-access-rights)。  [in] lpServiceName  要安装的服务的名称。 最大字符串长度为 256 个字符。 服务控制管理器数据库保留字符大小写，但服务名称比较始终不区分大小写。 正斜杠 (/) 和反斜杠 (\) 不是有效的服务名称字符。  [in, optional] lpDisplayName  用户界面程序用于标识服务的显示名称。 此字符串的最大长度为 256 个字符。 名称在服务控制管理器中保留大小写。 显示名称比较始终不区分大小写。  [in] dwDesiredAccess  对服务的访问权限。 在授予请求的访问权限之前，系统会检查调用进程的访问令牌。 有关值列表，请参阅 [服务安全性和访问权限](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-security-and-access-rights)。  //  // Service object specific access type  //  #define SERVICE\_QUERY\_CONFIG 0x0001  #define SERVICE\_CHANGE\_CONFIG 0x0002  #define SERVICE\_QUERY\_STATUS 0x0004  #define SERVICE\_ENUMERATE\_DEPENDENTS 0x0008  #define SERVICE\_START 0x0010  #define SERVICE\_STOP 0x0020  #define SERVICE\_PAUSE\_CONTINUE 0x0040  #define SERVICE\_INTERROGATE 0x0080  #define SERVICE\_USER\_DEFINED\_CONTROL 0x0100  #define SERVICE\_ALL\_ACCESS (STANDARD\_RIGHTS\_REQUIRED | \  SERVICE\_QUERY\_CONFIG | \  SERVICE\_CHANGE\_CONFIG | \  SERVICE\_QUERY\_STATUS | \  SERVICE\_ENUMERATE\_DEPENDENTS | \  SERVICE\_START | \  SERVICE\_STOP | \  SERVICE\_PAUSE\_CONTINUE | \  SERVICE\_INTERROGATE | \  SERVICE\_USER\_DEFINED\_CONTROL)  [in] dwServiceType  服务类型。 此参数的取值可为下列值之一：  展开表   |  |  | | --- | --- | | **值** | **含义** | | **SERVICE\_ADAPTER**  0x00000004 | 保留。 | | **SERVICE\_FILE\_SYSTEM\_DRIVER**  0x00000002 | 文件系统驱动程序服务。 | | **SERVICE\_KERNEL\_DRIVER**  0x00000001 | 驱动程序服务。 | | **SERVICE\_RECOGNIZER\_DRIVER**  0x00000008 | 保留。 | | **SERVICE\_WIN32\_OWN\_PROCESS**  0x00000010 | 在其自己的进程中运行的服务。 | | **SERVICE\_WIN32\_SHARE\_PROCESS**  0x00000020 | 与一个或多个其他服务共享进程的服务。 有关详细信息，请参阅 [服务程序](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-programs)。 |     如果指定 **SERVICE\_WIN32\_OWN\_PROCESS** 或 **SERVICE\_WIN32\_SHARE\_PROCESS**，并且服务在 [LocalSystem 帐户](https://learn.microsoft.com/zh-cn/windows/desktop/Services/localsystem-account)的上下文中运行，则还可以指定以下值。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **SERVICE\_INTERACTIVE\_PROCESS**  0x00000100 | 该服务可以与桌面交互。  有关详细信息，请参阅 [Interactive Services](https://learn.microsoft.com/zh-cn/windows/desktop/Services/interactive-services)。 |   [in] dwStartType  服务启动选项。 此参数的取值可为下列值之一：  展开表   |  |  | | --- | --- | | **值** | **含义** | | **SERVICE\_AUTO\_START**  0x00000002 | 服务控制管理器在系统启动期间自动启动的服务。 有关详细信息，请参阅 [自动启动服务](https://learn.microsoft.com/zh-cn/windows/desktop/Services/automatically-starting-services)。 | | **SERVICE\_BOOT\_START**  0x00000000 | 由系统加载程序启动的设备驱动程序。 此值只对驱动程序服务有效。 | | **SERVICE\_DEMAND\_START**  0x00000003 | 服务控制管理器在进程调用 [StartService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-startservicea) 函数时启动的服务。 有关详细信息，请参阅 [启动按需服务](https://learn.microsoft.com/zh-cn/windows/desktop/Services/starting-services-on-demand)。 | | **SERVICE\_DISABLED**  0x00000004 | 无法启动的服务。 尝试启动服务会导致错误代码 **ERROR\_SERVICE\_DISABLED**。 | | **SERVICE\_SYSTEM\_START**  0x00000001 | 由 **IoInitSystem** 函数启动的设备驱动程序。 此值只对驱动程序服务有效。 |   [in] dwErrorControl  错误的严重性，以及此服务无法启动时采取的操作。 此参数的取值可为下列值之一：  展开表   |  |  | | --- | --- | | **值** | **含义** | | **SERVICE\_ERROR\_CRITICAL**  0x00000003 | 如果可能，启动程序会将错误记录在事件日志中。 如果最近一次的正确配置正在启动，则启动操作会失败。 否则，系统会使用上一个已知正常的配置重启系统。 | | **SERVICE\_ERROR\_IGNORE**  0x00000000 | 启动程序将忽略错误并继续启动操作。 | | **SERVICE\_ERROR\_NORMAL**  0x00000001 | 启动程序将错误记录在事件日志中，但将继续执行启动操作。 | | **SERVICE\_ERROR\_SEVERE**  0x00000002 | 启动程序将错误记录在事件日志中。 如果正在启动最后一个已知良好的配置，则启动操作将继续。 否则，系统会使用最后一个已知良好的配置重启系统。 |   [in, optional] lpBinaryPathName  服务二进制文件的完全限定路径。 如果路径包含空格，则必须用引号将其括起来，以便正确解释该路径。 例如，“d：\my share\myservice.exe”应指定为“d：\my share\myservice.exe”。  路径还可以包含自动启动服务的参数。 例如，“d:\myshare\myservice.exe arg1 arg2”。 这些参数通常传递到服务入口点 (**main**函数) 。  如果在另一台计算机上指定路径，则共享必须由本地计算机的计算机帐户访问，因为这是远程调用中使用的安全上下文。 但是，此要求允许远程计算机中的任何潜在漏洞影响本地计算机。 因此，最好使用本地文件。  [in, optional] lpLoadOrderGroup  此服务所属的加载排序组的名称。 如果服务不属于组，请指定 NULL 或空字符串。  启动程序使用负载排序组，以相对于其他组的指定顺序加载服务组。 加载排序组的列表包含在以下注册表值中： **HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\ServiceGroupOrder**  [out, optional] lpdwTagId  指向变量的指针，该变量接收在 *lpLoadOrderGroup* 参数中指定的组中唯一的标记值。 如果未更改现有标记，请指定 NULL。  可以通过在以下注册表值中指定标记顺序向量，将标记用于对负载排序组中的服务启动进行排序：**HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\GroupOrderList**  仅针对具有 **SERVICE\_BOOT\_START** 或 **SERVICE\_SYSTEM\_START** 启动类型的驱动程序服务评估标记。  [in, optional] lpDependencies  指向以 null 分隔的服务名称或加载排序组的双 null 终止数组的指针，系统必须在此服务之前启动这些名称。 如果服务没有依赖项，请指定 NULL 或空字符串。 依赖于某个组意味着，如果在尝试启动该组的所有成员之后至少有一个组成员正在运行，则此服务可以运行。  必须为组名称添加 **SC\_GROUP\_IDENTIFIER** 前缀，以便可以将其与服务名称区分开来，因为服务和服务组共享相同的名称空间。  [in, optional] lpServiceStartName  运行服务的帐户的名称。 如果服务类型为SERVICE\_WIN32\_OWN\_PROCESS，请使用*域名*\用户名形式的帐户*名称*。 服务进程将作为此用户登录。 如果帐户属于内置域，则可以指定 .\*UserName*。  如果此参数为 **NULL，CreateService** 将使用 [LocalSystem 帐户](https://learn.microsoft.com/zh-cn/windows/desktop/Services/localsystem-account)。 如果服务类型指定 **SERVICE\_INTERACTIVE\_PROCESS**，则服务必须在 LocalSystem 帐户中运行。  如果此参数为 NT AUTHORITY\LocalService， **则 CreateService** 将使用 [LocalService 帐户](https://learn.microsoft.com/zh-cn/windows/desktop/Services/localservice-account)。 如果参数为 NT AUTHORITY\NetworkService， **则 CreateService** 将使用 [NetworkService 帐户](https://learn.microsoft.com/zh-cn/windows/desktop/Services/networkservice-account)。  共享进程可以以任何用户身份运行。  如果服务类型 **SERVICE\_KERNEL\_DRIVER** 或 **SERVICE\_FILE\_SYSTEM\_DRIVER**，则名称是系统用于加载设备驱动程序的驱动程序对象名称。 如果驱动程序要使用 I/O 系统创建的默认对象名称，请指定 NULL。  可以将服务配置为使用托管帐户或虚拟帐户。 如果服务配置为使用托管服务帐户，则名称为托管服务帐户名称。 如果服务配置为使用虚拟帐户，请将名称指定为 NT SERVICE\*ServiceName*。 有关托管服务帐户和虚拟帐户的详细信息，请参阅 [服务帐户分步指南](https://learn.microsoft.com/zh-cn/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/dd548356(v=ws.10))。  **Windows Server 2008、Windows Vista、Windows Server 2003 和 Windows XP：**在 Windows 7 和 Windows Server 2008 R2 之前，不支持托管服务帐户和虚拟帐户。  [in, optional] lpPassword  *lpServiceStartName* 参数指定的帐户名的密码。 如果帐户没有密码，或者服务在 LocalService、NetworkService 或 LocalSystem 帐户中运行，请指定一个空字符串。 有关详细信息，请参阅 [服务记录列表](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-record-list)。  如果 *lpServiceStartName* 参数指定的帐户名称是托管服务帐户的名称或虚拟帐户名称， *则 lpPassword* 参数必须为 NULL。  对于驱动程序服务，将忽略密码。 返回值 如果函数成功，则返回值是服务的句柄。  如果函数失败，则返回值为 NULL。 要获得更多的错误信息，请调用 GetLastError。  服务控制管理器可以设置以下错误代码。 其他错误代码可由服务控制管理器调用的注册表函数设置。  展开表   |  |  | | --- | --- | | **返回代码** | **说明** | | **ERROR\_ACCESS\_DENIED** | SCM 数据库的句柄没有 **SC\_MANAGER\_CREATE\_SERVICE** 访问权限。 | | **ERROR\_CIRCULAR\_DEPENDENCY** | 指定了循环服务依赖项。 | | **ERROR\_DUPLICATE\_SERVICE\_NAME** | 显示名称已作为服务名称或另一个显示名称存在于服务控制管理器数据库中。 | | **ERROR\_INVALID\_HANDLE** | 指定服务控制管理器数据库的句柄无效。 | | **ERROR\_INVALID\_NAME** | 指定的服务名称无效。 | | **ERROR\_INVALID\_PARAMETER** | 指定的参数无效。 | | **ERROR\_INVALID\_SERVICE\_ACCOUNT** | *lpServiceStartName* 参数中指定的用户帐户名称不存在。 | | **ERROR\_SERVICE\_EXISTS** | 指定的服务已存在于此数据库中。 | | **ERROR\_SERVICE\_MARKED\_FOR\_DELETE** | 指定的服务已存在于此数据库中，并且已标记为要删除。 |  注解 **CreateService** 函数创建一个服务对象，并通过在以下注册表项下创建一个与服务同名的项，将其安装在服务控制管理器数据库中：**HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services**  **由 CreateService**、[ChangeServiceConfig](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-changeserviceconfiga) 和 [ChangeServiceConfig2](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-changeserviceconfig2a) 指定的信息将保存为此键下的值。 下面是为服务存储的值的示例。  展开表   |  |  | | --- | --- | | **值** | **说明** | | **DependOnGroup** | 此服务所依赖的负载排序组，由 *lpDependencies* 指定。 | | **DependOnService** | 此服务所依赖的服务，由 *lpDependencies* 指定。 | | **说明** | [由 ChangeServiceConfig2](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-changeserviceconfig2a) 指定的说明。 | | **DisplayName** | *由 lpDisplayName* 指定的显示名称。 | | **ErrorControl** | *dwErrorControl* 指定的错误控制。 | | **FailureActions** | [ChangeServiceConfig2](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-changeserviceconfig2a) 指定的失败操作。 | | **组** | *由 lpLoadOrderGroup* 指定的加载排序组。 请注意，设置此值可以替代 **DependOnService** 值的设置。 | | **ImagePath** | 二进制文件的名称，由 *lpBinaryPathName 指定*。 | | **ObjectName** | 由 *lpServiceStartName* 指定的帐户名称。 | | **启动** | 何时启动服务，由 *dwStartType 指定*。 | | **标记** | *由 lpdwTagId* 指定的标记标识符。 | | 类型 | *由 dwServiceType* 指定的服务类型。 |     安装程序和服务本身可以为特定于服务的信息创建其他子项。  返回的句柄仅对调用 **CreateService** 的进程有效。 可以通过调用 [CloseServiceHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-closeservicehandle) 函数将其关闭。  如果要创建共享进程的服务，请避免调用具有进程范围效果的函数，例如 [ExitProcess](https://learn.microsoft.com/zh-cn/windows/desktop/api/processthreadsapi/nf-processthreadsapi-exitprocess)。 此外，请勿卸载服务 DLL。 示例 有关示例，请参阅 [安装服务](https://learn.microsoft.com/zh-cn/windows/desktop/Services/installing-a-service)。 要求  |  |  | | --- | --- | | **标头** | winsvc.h (包括 Windows.h) | | **Library** | Advapi32.lib | | **DLL** | Advapi32.dll |  另请参阅 [ChangeServiceConfig](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-changeserviceconfiga)  [ChangeServiceConfig2](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-changeserviceconfig2a)  [CloseServiceHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-closeservicehandle)  [ControlService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-controlservice)  [DeleteService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-deleteservice)  [EnumDependentServices](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-enumdependentservicesa)  [OpenSCManager](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openscmanagera)  [QueryServiceConfig](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryserviceconfiga)  [QueryServiceDynamicInformation](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryservicedynamicinformation)  [QueryServiceObjectSecurity](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryserviceobjectsecurity)  [QueryServiceStatusEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryservicestatusex)  [服务帐户分步指南](https://learn.microsoft.com/zh-cn/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/dd548356(v=ws.10))  [服务函数](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-functions)  [服务安装、删除和枚举](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-installation-removal-and-enumeration)  [SetServiceObjectSecurity](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-setserviceobjectsecurity)  [StartService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-startservicea) |

## DeleteService函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 将指定的服务标记为从服务控制管理器数据库中删除。 语法 C++  BOOL DeleteService(  [in] SC\_HANDLE hService  ); 参数 [in] hService  服务的句柄。 此句柄由 [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea) 或 [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea) 函数返回，并且必须具有 DELETE 访问权限。 有关详细信息，请参阅 [服务安全和访问权限](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-security-and-access-rights)。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 要获得更多的错误信息，请调用 GetLastError。  以下错误代码可能由服务控制管理器设置。 其他可能由服务控制管理器调用的注册表函数设置。  展开表   |  |  | | --- | --- | | **返回代码** | **说明** | | **ERROR\_ACCESS\_DENIED** | 句柄没有 DELETE 访问权限。 | | **ERROR\_INVALID\_HANDLE** | 指定的句柄无效。 | | **ERROR\_SERVICE\_MARKED\_FOR\_DELETE** | 指定的服务已标记为要删除。 |  注解 **DeleteService** 函数将服务标记为要从服务控制管理器数据库中删除。 在通过调用 [CloseServiceHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-closeservicehandle) 函数关闭服务的所有打开句柄并且服务未运行之前，不会删除数据库条目。 通过使用SERVICE\_CONTROL\_STOP控件代码调用 [ControlService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-controlservice) 函数来停止正在运行的服务。 如果服务无法停止，则重启系统时会删除数据库条目。  服务控制管理器通过从注册表中删除服务键及其子项来删除服务。 示例 有关示例，请参阅 [删除服务](https://learn.microsoft.com/zh-cn/windows/desktop/Services/deleting-a-service)。 要求  |  |  | | --- | --- | | **标头** | winsvc.h (包括 Windows.h) | | **Library** | Advapi32.lib | | **DLL** | Advapi32.dll |  另请参阅 [CloseServiceHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-closeservicehandle)  [ControlService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-controlservice)  [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea)  [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea)  [服务函数](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-functions)  [服务安装、删除和枚举](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-installation-removal-and-enumeration) |

## OpenService函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 打开现有服务。 语法 C++复制  SC\_HANDLE OpenServiceA(  [in] SC\_HANDLE hSCManager,  [in] LPCSTR lpServiceName,  [in] DWORD dwDesiredAccess  ); 参数 [in] hSCManager  服务控制管理器数据库的句柄。 [OpenSCManager](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openscmanagera) 函数返回此句柄。 有关详细信息，请参阅 [服务安全和访问权限](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-security-and-access-rights)。  [in] lpServiceName  要打开的服务的名称。 这是创建服务对象时 CreateService 函数的 *lpServiceName* 参数指定的名称，而不是用户界面应用程序为标识服务而显示的服务显示名称。  最大字符串长度为 256 个字符。 服务控制管理器数据库保留字符的大小写，但服务名称比较始终不区分大小写。 正斜杠 (/) 和反斜杠 (\) 是无效的服务名称字符。  [in] dwDesiredAccess  对服务的访问权限。 有关访问权限的列表，请参阅 [服务安全性和访问权限](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-security-and-access-rights)。  在授予请求的访问权限之前，系统会根据与服务对象关联的安全描述符的任意访问控制列表检查调用进程的访问令牌。 返回值 如果函数成功，则返回值是服务的句柄。  如果函数失败，则返回值为 NULL。 要获得更多的错误信息，请调用 GetLastError。  服务控制管理器可以设置以下错误代码。 其他可由服务控制管理器调用的注册表函数设置。  展开表   |  |  | | --- | --- | | **返回代码** | **说明** | | **ERROR\_ACCESS\_DENIED** | 句柄无权访问服务。 | | **ERROR\_INVALID\_HANDLE** | 指定的句柄无效。 | | **ERROR\_INVALID\_NAME** | 指定的服务名称无效。 | | **ERROR\_SERVICE\_DOES\_NOT\_EXIST** | 指定的服务不存在。 |  注解 返回的句柄仅对调用 **OpenService** 的进程有效。 可以通过调用 [CloseServiceHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-closeservicehandle) 函数来关闭它。  若要使用 **OpenService**，除了 **SC\_MANAGER\_CONNECT**之外，不需要任何特权。 示例 有关示例，请参阅 [启动服务](https://learn.microsoft.com/zh-cn/windows/desktop/Services/starting-a-service)。  **备注**  winsvc.h 标头将 OpenService 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 要求  |  |  | | --- | --- | | **目标平台** | Windows | | **标头** | winsvc.h (包括 Windows.h) | | **Library** | Advapi32.lib | | **DLL** | Advapi32.dll |  另请参阅 [ChangeServiceConfig](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-changeserviceconfiga)  [CloseServiceHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-closeservicehandle)  [ControlService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-controlservice)  [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea)  [DeleteService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-deleteservice)  [EnumDependentServices](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-enumdependentservicesa)  [OpenSCManager](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openscmanagera)  [QueryServiceConfig](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryserviceconfiga)  [QueryServiceDynamicInformation](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryservicedynamicinformation)  [QueryServiceObjectSecurity](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryserviceobjectsecurity)  [QueryServiceStatusEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryservicestatusex)  [SCM 句柄](https://learn.microsoft.com/zh-cn/windows/desktop/Services/scm-handles)  [服务函数](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-functions)  [SetServiceObjectSecurity](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-setserviceobjectsecurity)  [StartService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-startservicea) |

## StartService函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 启动服务。 语法 C++复制  BOOL StartServiceA(  [in] SC\_HANDLE hService,  [in] DWORD dwNumServiceArgs,  [in, optional] LPCSTR \*lpServiceArgVectors  ); 参数 [in] hService  服务的句柄。 此句柄由 [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea) 或 [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea) 函数返回，并且必须具有SERVICE\_START访问权限。 有关详细信息，请参阅 [服务安全和访问权限](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-security-and-access-rights)。  [in] dwNumServiceArgs  *lpServiceArgVectors* 数组中的字符串数。 如果 *lpServiceArgVectors* 为 NULL，则此参数可以为零。  [in, optional] lpServiceArgVectors  要作为参数传递给 [服务的 ServiceMain](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nc-winsvc-lpservice_main_functiona) 函数的以 null 结尾的字符串。 如果没有参数，此参数可以为 NULL。 否则， (lpServiceArgVectors[0]) 的第一个参数是服务的名称，后跟 (lpServiceArgVectors[1] 到 lpServiceArgVectors[dwNumServiceArgs-1] ) 的任何其他参数。  驱动程序服务不会接收这些参数。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 要获得更多的错误信息，请调用 GetLastError。  服务控制管理器可以设置以下错误代码。 其他可由服务控制管理器调用的注册表函数设置。  展开表   |  |  | | --- | --- | | **返回代码** | **说明** | | **ERROR\_ACCESS\_DENIED** | 句柄没有SERVICE\_START访问权限。 | | **ERROR\_INVALID\_HANDLE** | 该句柄无效。 | | **ERROR\_PATH\_NOT\_FOUND** | 找不到服务二进制文件。 | | **ERROR\_SERVICE\_ALREADY\_RUNNING** | 服务的实例已在运行。 | | **ERROR\_SERVICE\_DATABASE\_LOCKED** | 数据库已锁定。 | | **ERROR\_SERVICE\_DEPENDENCY\_DELETED** | 该服务依赖于不存在或已标记为删除的服务。 | | **ERROR\_SERVICE\_DEPENDENCY\_FAIL** | 该服务依赖于另一个无法启动的服务。 | | **ERROR\_SERVICE\_DISABLED** | 服务已被禁用。 | | **ERROR\_SERVICE\_LOGON\_FAILED** | 由于登录失败而无法启动服务。 如果将服务配置为在没有“作为服务登录”权限的帐户下运行，则会发生此错误。 | | **ERROR\_SERVICE\_MARKED\_FOR\_DELETE** | 服务已标记为要删除。 | | **ERROR\_SERVICE\_NO\_THREAD** | 无法为服务创建线程。 | | **ERROR\_SERVICE\_REQUEST\_TIMEOUT** | 服务的进程已启动，但它未调用 [StartServiceCtrlDispatcher](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-startservicectrldispatchera)，或者调用 **StartServiceCtrlDispatcher** 的线程可能在控制处理程序函数中被阻止。 |  注解 启动驱动程序服务时，在设备驱动程序完成初始化之前， **StartService** 函数不会返回。  启动服务时，服务控制管理器 (SCM) 生成服务进程（如有必要）。 如果指定的服务与其他服务共享进程，则所需的进程可能已存在。 **StartService** 函数不会等待新服务的第一次状态更新，因为它可能需要一段时间。 相反，当 SCM 收到来自服务控制调度程序的通知，指示已成功创建此服务的 [ServiceMain](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nc-winsvc-lpservice_main_functiona) 线程时，它将返回 。  SCM 在从 **StartService** 返回之前设置以下默认状态值：   * 服务的当前状态设置为SERVICE\_START\_PENDING。 * 接受的控件设置为无 (零) 。 * CheckPoint 值设置为零。 * WaitHint 时间设置为 2 秒。   调用进程可以通过定期调用 [QueryServiceStatus](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryservicestatus) 函数来查询服务的状态来确定新服务是否已完成初始化。  服务在初始化期间无法调用 **StartService** 。 原因是 SCM 在初始化期间锁定服务控制数据库，因此对 **StartService** 的调用将受阻。 服务向 SCM 报告已成功启动后，可以调用 **StartService**。  与 [ControlService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-controlservice) 一样，如果任何服务忙于处理控制代码， **StartService** 将阻塞 30 秒。 如果超时过期时忙服务仍未从其处理程序函数返回， **则 StartService** 将失败并ERROR\_SERVICE\_REQUEST\_TIMEOUT。 这是因为 SCM 一次只处理一个服务控制通知。 示例 有关示例，请参阅 [启动服务](https://learn.microsoft.com/zh-cn/windows/desktop/Services/starting-a-service)。  **备注**  winsvc.h 标头将 StartService 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 要求  |  |  | | --- | --- | | **标头** | winsvc.h (包括 Windows.h) | | **Library** | Advapi32.lib | | **DLL** | Advapi32.dll |  另请参阅 [ControlService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-controlservice)  [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea)  [DeleteService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-deleteservice)  [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea)  [QueryServiceDynamicInformation](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryservicedynamicinformation)  [QueryServiceStatusEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryservicestatusex)  [服务函数](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-functions)  [服务启动](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-startup)  [ServiceMain](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nc-winsvc-lpservice_main_functiona) |

## ControlService函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 将控制代码发送到服务。  若要在停止服务时指定其他信息，请使用 [ControlServiceEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-controlserviceexa) 函数。 语法 C++复制  BOOL ControlService(  [in] SC\_HANDLE hService,  [in] DWORD dwControl,  [out] LPSERVICE\_STATUS lpServiceStatus  ); 参数 [in] hService  服务的句柄。 此句柄由 [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea) 或 [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea) 函数返回。 此句柄所需的 [访问权限](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-security-and-access-rights) 取决于请求的 *dwControl* 代码。  [in] dwControl  此参数可以是以下控制代码之一。  展开表   |  |  | | --- | --- | | **控制代码** | **含义** | | **SERVICE\_CONTROL\_CONTINUE**  0x00000003 | 通知暂停的服务应恢复。 *hService* 句柄必须具有**SERVICE\_PAUSE\_CONTINUE**访问权限。 | | **SERVICE\_CONTROL\_INTERROGATE**  0x00000004 | 通知服务它应将其当前状态信息报告给服务控制管理器。 *hService* 句柄必须具有**SERVICE\_INTERROGATE**访问权限。  请注意，此控件通常并不有用，因为 SCM 知道服务的当前状态。 | | **SERVICE\_CONTROL\_NETBINDADD**  0x00000007 | 通知网络服务有用于绑定的新组件。 *hService* 句柄必须具有**SERVICE\_PAUSE\_CONTINUE**访问权限。 但是，此控制代码已弃用;请改用即插即用功能。 | | **SERVICE\_CONTROL\_NETBINDDISABLE**  0x0000000A | 通知网络服务其其中一个绑定已被禁用。 *hService* 句柄必须具有**SERVICE\_PAUSE\_CONTINUE**访问权限。 但是，此控制代码已弃用;请改用即插即用功能。 | | **SERVICE\_CONTROL\_NETBINDENABLE**  0x00000009 | 通知网络服务已启用禁用的绑定。 *hService* 句柄必须具有**SERVICE\_PAUSE\_CONTINUE**访问权限。 但是，此控制代码已弃用;请改用即插即用功能。 | | **SERVICE\_CONTROL\_NETBINDREMOVE**  0x00000008 | 通知网络服务已删除用于绑定的组件。 *hService* 句柄必须具有**SERVICE\_PAUSE\_CONTINUE**访问权限。 但是，此控制代码已弃用;请改用即插即用功能。 | | **SERVICE\_CONTROL\_PARAMCHANGE**  0x00000006 | 通知服务其启动参数已更改。 *hService* 句柄必须具有**SERVICE\_PAUSE\_CONTINUE**访问权限。 | | **SERVICE\_CONTROL\_PAUSE**  0x00000002 | 通知服务它应暂停。 *hService* 句柄必须具有**SERVICE\_PAUSE\_CONTINUE**访问权限。 | | **SERVICE\_CONTROL\_STOP**  0x00000001 | 通知服务它应停止。 *hService* 句柄必须具有**SERVICE\_STOP**访问权限。  将停止请求发送到服务后，不应向服务发送其他控件。 |     此值也可以是用户定义的控件代码，如下表所述。  展开表   |  |  | | --- | --- | | **控制代码** | **含义** | | 范围 128 到 255 | 服务定义与控件代码关联的操作。 *hService* 句柄必须具有**SERVICE\_USER\_DEFINED\_CONTROL**访问权限。 |   [out] lpServiceStatus  指向接收最新服务状态信息的 [SERVICE\_STATUS](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/ns-winsvc-service_status) 结构的指针。 返回的信息反映了服务向服务控制管理器报告的最新状态。  仅当 [GetLastError](https://learn.microsoft.com/zh-cn/windows/win32/api/errhandlingapi/nf-errhandlingapi-getlasterror) 返回以下错误代码之一时，服务控制管理器才会填充结构： **NO\_ERROR**、 **ERROR\_INVALID\_SERVICE\_CONTROL**、 **ERROR\_SERVICE\_CANNOT\_ACCEPT\_CTRL**或 **ERROR\_SERVICE\_NOT\_ACTIVE**。 否则，不会填充结构。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 要获得更多的错误信息，请调用 GetLastError。  服务控制管理器可以设置以下错误代码。 其他错误代码可由服务控制管理器调用的注册表函数设置。  展开表   |  |  | | --- | --- | | **返回代码** | **说明** | | **ERROR\_ACCESS\_DENIED** | 句柄没有所需的访问权限。 | | **ERROR\_DEPENDENT\_SERVICES\_RUNNING** | 无法停止该服务，因为其他正在运行的服务都依赖于它。 | | **ERROR\_INVALID\_HANDLE** | 未使用 [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea) 或 [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea) 获取指定的句柄，或者该句柄不再有效。 | | **ERROR\_INVALID\_PARAMETER** | 请求的控制代码未定义。 | | **ERROR\_INVALID\_SERVICE\_CONTROL** | 请求的控制代码无效或服务无法接受该控制代码。 | | **ERROR\_SERVICE\_CANNOT\_ACCEPT\_CTRL** | 无法将请求的控制代码发送到服务，因为服务的状态为 **SERVICE\_STOPPED**、 **SERVICE\_START\_PENDING**或 **SERVICE\_STOP\_PENDING**。 | | **ERROR\_SERVICE\_NOT\_ACTIVE** | 该服务尚未启动。 | | **ERROR\_SERVICE\_REQUEST\_TIMEOUT** | 服务的进程已启动，但它未调用 [StartServiceCtrlDispatcher](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-startservicectrldispatchera)，或者调用 **StartServiceCtrlDispatcher** 的线程可能会在控件处理程序函数中被阻止。 | | **ERROR\_SHUTDOWN\_IN\_PROGRESS** | 系统正在关闭。 |  注解 **ControlService** 函数要求服务控制管理器 (SCM) 将请求的控制代码发送到服务。 如果服务已指定它将接受代码，并且处于可向其发送控制代码的状态，则 SCM 会发送代码。  SCM 以串行方式处理服务控制通知，它将等待一个服务完成服务控制通知的处理，然后再发送下一个服务控制通知。 因此，如果任何服务忙于处理控件代码，则对 **ControlService** 的调用将阻止 30 秒。 如果在超时过期时忙服务仍未从其处理程序函数返回， **则 ControlService** 将失败并 **ERROR\_SERVICE\_REQUEST\_TIMEOUT**。  若要停止和启动服务，需要一个允许执行此操作的安全描述符。 默认安全描述符允许 [LocalSystem 帐户](https://learn.microsoft.com/zh-cn/windows/desktop/Services/localsystem-account)以及管理员组和 Power Users 组的成员停止和启动服务。 若要更改服务的安全描述符，请参阅 [修改服务的 DACL](https://learn.microsoft.com/zh-cn/windows/desktop/Services/modifying-the-dacl-for-a-service)。  [QueryServiceStatusEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryservicestatusex) 函数返回[一个SERVICE\_STATUS\_PROCESS](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/ns-winsvc-service_status_process)结构，其 **dwCurrentState** 和 **dwControlsAccepted** 成员指示正在运行的服务接受的当前状态和控件。 默认情况下，所有正在运行的服务都接受 **SERVICE\_CONTROL\_INTERROGATE** 控制代码。 驱动程序不接受 **除SERVICE\_CONTROL\_STOP** 和 **SERVICE\_CONTROL\_INTERROGATE**以外的控制代码。 每个服务指定它在调用 [SetServiceStatus](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-setservicestatus) 函数以报告其状态时接受的其他控制代码。 服务在运行时应始终接受这些代码，无论它正在执行什么操作。  下表显示了 SCM 在每个可能的服务状态中的操作。  展开表   |  |  |  | | --- | --- | --- | | **服务状态** | **停止** | **其他控件** | | STOPPED | (c) | (c) | | **STOP\_PENDING** | (b) | (b) | | **START\_PENDING** | () | (b) | | **运行** | () | () | | **CONTINUE\_PENDING** | () | () | | **PAUSE\_PENDING** | () | () | | **暂停** | () | () |   ()  如果服务接受此控制代码，请将请求发送到服务;否则， **ControlService** 返回零， [GetLastError](https://learn.microsoft.com/zh-cn/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror) 返回 **ERROR\_INVALID\_SERVICE\_CONTROL**。  (b)  服务不处于可向其发送控件的状态，因此 **ControlService** 返回零， [GetLastError](https://learn.microsoft.com/zh-cn/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror) 返回 **ERROR\_SERVICE\_CANNOT\_ACCEPT\_CTRL**。  (c)  服务未处于活动状态，因此 **ControlService** 返回零， [GetLastError](https://learn.microsoft.com/zh-cn/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror) 返回 **ERROR\_SERVICE\_NOT\_ACTIVE**。 示例 有关示例，请参阅 [停止服务](https://learn.microsoft.com/zh-cn/windows/desktop/Services/stopping-a-service)。 要求  |  |  | | --- | --- | | **标头** | winsvc.h (包括 Windows.h) | | **Library** | AdvApi32.lib | | **DLL** | AdvApi32.dll |  另请参阅 [ControlServiceEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-controlserviceexa)  [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea)  [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea)  [QueryServiceStatusEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryservicestatusex)  [SERVICE\_STATUS](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/ns-winsvc-service_status)  [服务控制请求](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-control-requests)  [服务函数](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-functions)  [SetServiceObjectSecurity](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-setserviceobjectsecurity)  [SetServiceStatus](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-setservicestatus) |

## QueryServiceStatus函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 检索指定服务的当前状态。  此函数已被 [QueryServiceStatusEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryservicestatusex) 函数取代。 **QueryServiceStatusEx** 返回 **QueryServiceStatus** 返回的相同信息，同时添加了服务的进程标识符和其他信息。 语法 C++复制  BOOL QueryServiceStatus(  [in] SC\_HANDLE hService,  [out] LPSERVICE\_STATUS lpServiceStatus  ); 参数 [in] hService  服务的句柄。 此句柄由 [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea) 或 [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea) 函数返回，并且它必须具有SERVICE\_QUERY\_STATUS访问权限。 有关详细信息，请参阅 [服务安全和访问权限](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-security-and-access-rights)。  [out] lpServiceStatus  指向接收状态信息的 [SERVICE\_STATUS](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/ns-winsvc-service_status) 结构的指针。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 要获得更多的错误信息，请调用 GetLastError。  服务控制管理器可以设置以下错误代码。 其他错误代码可由服务控制管理器调用的注册表函数设置。  展开表   |  |  | | --- | --- | | **返回代码** | **说明** | | **ERROR\_ACCESS\_DENIED** | 句柄没有SERVICE\_QUERY\_STATUS访问权限。 | | **ERROR\_INVALID\_HANDLE** | 该句柄无效。 |  注解 **QueryServiceStatus** 函数返回向服务控制管理器报告的最新服务状态信息。 如果服务刚刚更改了其状态，则可能尚未更新服务控制管理器。 要求  |  |  | | --- | --- | | **标头** | winsvc.h (包括 Windows.h) | | **Library** | Advapi32.lib | | **DLL** | Advapi32.dll |  另请参阅 [ControlService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-controlservice)  [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea)  [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea)  [QueryServiceStatusEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryservicestatusex)  [SERVICE\_STATUS](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/ns-winsvc-service_status)  [服务函数](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-functions)  [服务启动](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-startup)  [SetServiceStatus](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-setservicestatus) |

## QueryServiceStatusEx 函数

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 根据指定的信息级别检索指定服务的当前状态。 语法 C++复制  BOOL QueryServiceStatusEx(  [in] SC\_HANDLE hService,  [in] SC\_STATUS\_TYPE InfoLevel,  [out, optional] LPBYTE lpBuffer,  [in] DWORD cbBufSize,  [out] LPDWORD pcbBytesNeeded  ); 参数 [in] hService  服务的句柄。 此句柄由 [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea) 或 [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea) 函数返回，并且它必须具有SERVICE\_QUERY\_STATUS访问权限。 有关详细信息，请参阅 [服务安全性和访问权限](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-security-and-access-rights)。  [in] InfoLevel  要返回的服务属性。 使用 SC\_STATUS\_PROCESS\_INFO 检索服务状态信息。 *lpBuffer* 参数是指向[SERVICE\_STATUS\_PROCESS](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/ns-winsvc-service_status_process)结构的指针。  目前，未定义其他信息级别。  [out, optional] lpBuffer  指向接收状态信息的缓冲区的指针。 此数据的格式取决于 *InfoLevel* 参数的值。  此数组的最大大小为 8K 字节。 若要确定所需的大小，请为此参数指定 NULL，为 *cbBufSize* 参数指定 0。 函数将失败， [GetLastError](https://learn.microsoft.com/zh-cn/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror) 将返回ERROR\_INSUFFICIENT\_BUFFER。 *“线路板”参数*将接收所需大小。  [in] cbBufSize  *lpBuffer* 参数指向的缓冲区大小（以字节为单位）。  [out] pcbBytesNeeded  指向变量的指针，该变量接收存储所有状态信息所需的字节数（如果函数失败并ERROR\_INSUFFICIENT\_BUFFER）。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 要获得更多的错误信息，请调用 GetLastError。 可能会返回以下错误。  展开表   |  |  | | --- | --- | | **返回代码** | **说明** | | **ERROR\_INVALID\_HANDLE** | 该句柄无效。 | | **ERROR\_ACCESS\_DENIED** | 句柄没有SERVICE\_QUERY\_STATUS访问权限。 | | **ERROR\_INSUFFICIENT\_BUFFER** | 缓冲区太小， [无法](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/ns-winsvc-service_status_process) SERVICE\_STATUS\_PROCESS结构。 结构中未写入任何内容。 | | **ERROR\_INVALID\_PARAMETER** | [SERVICE\_STATUS\_PROCESS](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/ns-winsvc-service_status_process)的 **cbSize** 成员无效。 | | **ERROR\_INVALID\_LEVEL** | *InfoLevel* 参数包含不受支持的值。 | | **ERROR\_SHUTDOWN\_IN\_PROGRESS** | 系统正在关闭;无法调用此函数。 |  注解 **QueryServiceStatusEx** 函数返回报告给服务控制管理器的最新服务状态信息。 如果服务刚刚更改其状态，则它可能尚未更新服务控制管理器。  [如果服务](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/ns-winsvc-service_status_process)的状态是SERVICE\_RUNNING、SERVICE\_PAUSE\_PENDING、SERVICE\_PAUSED或SERVICE\_CONTINUE\_PENDING之一，则SERVICE\_STATUS\_PROCESS结构中返回的进程标识符是有效的。 但是，如果服务处于SERVICE\_START\_PENDING或SERVICE\_STOP\_PENDING状态，则进程标识符可能无效，并且如果服务处于SERVICE\_STOPPED状态，则它永远不会有效。 示例 有关示例，请参阅 [启动服务](https://learn.microsoft.com/zh-cn/windows/desktop/Services/starting-a-service) 或 [停止服务](https://learn.microsoft.com/zh-cn/windows/desktop/Services/stopping-a-service)。 要求  |  |  | | --- | --- | | **标头** | winsvc.h (包括 Windows.h) | | **Library** | Advapi32.lib | | **DLL** | Advapi32.dll |  另请参阅 [SERVICE\_STATUS\_PROCESS](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/ns-winsvc-service_status_process)  [服务函数](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-functions)  [服务启动](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-startup) |

## SetNamedSecurityInfoA 函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SetNamedSecurityInfo** 函数在指定对象[的安全描述符](https://learn.microsoft.com/zh-cn/windows/desktop/SecGloss/s-gly)中设置指定的安全信息。 调用方按名称标识对象。 语法 C++复制  DWORD SetNamedSecurityInfoA(  [in] LPSTR pObjectName,  [in] SE\_OBJECT\_TYPE ObjectType,  [in] SECURITY\_INFORMATION SecurityInfo,  [in, optional] PSID psidOwner,  [in, optional] PSID psidGroup,  [in, optional] PACL pDacl,  [in, optional] PACL pSacl  ); 参数 [in] pObjectName  指向 **以 null** 结尾的字符串的指针，该字符串指定要为其设置安全信息的对象的名称。 这可以是 NTFS 文件系统、网络共享、注册表项、信号灯、事件、互斥体、文件映射或可等待计时器上的本地或远程文件或目录的名称。  有关不同对象类型的字符串格式的说明，请参阅 [SE\_OBJECT\_TYPE](https://learn.microsoft.com/zh-cn/windows/desktop/api/accctrl/ne-accctrl-se_object_type)。  [in] ObjectType  [SE\_OBJECT\_TYPE](https://learn.microsoft.com/zh-cn/windows/desktop/api/accctrl/ne-accctrl-se_object_type)枚举的值，指示*由 pObjectName* 参数命名的对象的类型。  [in] SecurityInfo  一组位标志，指示要设置的安全信息的类型。 此参数可以是 [SECURITY\_INFORMATION](https://learn.microsoft.com/zh-cn/windows/desktop/SecAuthZ/security-information) 位标志的组合。  [in, optional] psidOwner  指向标识对象所有者的 [SID](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/ns-winnt-sid) 结构的指针。 如果调用方没有 **SeRestorePrivilege** 常量 (请参阅 [特权常量](https://learn.microsoft.com/zh-cn/windows/desktop/SecAuthZ/privilege-constants)) ，则此 **SID** 必须包含在调用方令牌中，并且必须启用 **SE\_GROUP\_OWNER** 权限。 *SecurityInfo* 参数必须包含 OWNER\_SECURITY\_INFORMATION 标志。 若要设置所有者，调用方必须具有对对象的WRITE\_OWNER访问权限或启用SE\_TAKE\_OWNERSHIP\_NAME特权。 如果未设置所有者 **SID**，则此参数可以为 **NULL**。  [in, optional] psidGroup  指向标识对象的主要组的 SID 的指针。 *SecurityInfo* 参数必须包含 GROUP\_SECURITY\_INFORMATION 标志。 如果未设置主组 SID，则此参数可以为 **NULL**。  [in, optional] pDacl  指向 对象的新 DACL 的指针。 *SecurityInfo* 参数必须包含 DACL\_SECURITY\_INFORMATION 标志。 调用方必须具有对 对象的WRITE\_DAC访问权限或对象的所有者。 如果未设置 DACL，则此参数可以为 **NULL**。  [in, optional] pSacl  指向 对象的新 SACL 的指针。 *SecurityInfo* 参数必须包含以下任何标志：SACL\_SECURITY\_INFORMATION、LABEL\_SECURITY\_INFORMATION、ATTRIBUTE\_SECURITY\_INFORMATION、SCOPE\_SECURITY\_INFORMATION或BACKUP\_SECURITY\_INFORMATION。  如果设置SACL\_SECURITY\_INFORMATION或SCOPE\_SECURITY\_INFORMATION，则调用方必须启用SE\_SECURITY\_NAME特权。 如果未设置 SACL，则此参数可以为 **NULL**。 返回值 如果函数成功，该函数将返回ERROR\_SUCCESS。  如果函数失败，它将返回 WinError.h 中定义的非零错误代码。 注解 如果要 (DACL) 或 (SACL) 对象的[系统访问控制列表中](https://learn.microsoft.com/zh-cn/windows/desktop/SecGloss/s-gly)的任何元素设置[任意](https://learn.microsoft.com/zh-cn/windows/desktop/SecGloss/d-gly)访问控制 (列表，系统会根据[继承规则](https://learn.microsoft.com/zh-cn/windows/desktop/SecAuthZ/ace-inheritance-rules)自动将) ) 的任何可继承[访问控制项](https://learn.microsoft.com/zh-cn/windows/desktop/SecGloss/a-gly)传播到现有子对象。  可以将 **SetNamedSecurityInfo** 函数用于以下类型的对象：   * NTFS 上的本地或远程文件或目录 * 本地或远程打印机 * 本地或远程 Windows 服务 * 网络共享 * 注册表项 * 信号灯、事件、互斥体和可等待计时器 * 文件映射对象 * 目录服务对象   **SetNamedSecurityInfo** 函数不会根据首选顺序重新排序允许访问或拒绝访问的 ACE。 将可继承的 ACE 传播到现有子对象时， **SetNamedSecurityInfo** 将继承的 ACE 按顺序放在子对象的 DACL 中的所有非继承 ACE 之后。  此函数以 [纯文本](https://learn.microsoft.com/zh-cn/windows/desktop/SecGloss/p-gly)形式传输信息。 除非系统已关闭签名，但不会执行加密，否则此函数传输的信息将进行签名。  更新由 UNC 路径指示的文件夹（例如 \Test\TestFolder）的访问权限时，将删除原始继承的 ACE，并且不包含完整卷路径。 示例 有关使用此函数的示例，请参阅 [修改对象的 ACL](https://learn.microsoft.com/zh-cn/windows/desktop/SecAuthZ/modifying-the-acls-of-an-object-in-c--) 或 [获取对象所有权](https://learn.microsoft.com/zh-cn/windows/desktop/SecAuthZ/taking-object-ownership-in-c--)。  **备注**  aclapi.h 标头将 SetNamedSecurityInfo 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名的使用与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 要求  |  |  | | --- | --- | | **标头** | aclapi.h | | **Library** | Advapi32.lib | | **DLL** | Advapi32.dll |  另请参阅 [ACL](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/ns-winnt-acl)  [访问控制](https://learn.microsoft.com/zh-cn/windows/desktop/SecAuthZ/access-control)  [基本访问控制函数](https://learn.microsoft.com/zh-cn/windows/desktop/SecAuthZ/authorization-functions)  [GetNamedSecurityInfo](https://learn.microsoft.com/zh-cn/windows/desktop/api/aclapi/nf-aclapi-getnamedsecurityinfoa)  [GetSecurityInfo](https://learn.microsoft.com/zh-cn/windows/desktop/api/aclapi/nf-aclapi-getsecurityinfo)  [SECURITY\_DESCRIPTOR](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/ns-winnt-security_descriptor)  [SECURITY\_INFORMATION](https://learn.microsoft.com/zh-cn/windows/desktop/SecAuthZ/security-information)  [SE\_OBJECT\_TYPE](https://learn.microsoft.com/zh-cn/windows/desktop/api/accctrl/ne-accctrl-se_object_type)  [SID](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/ns-winnt-sid)  [SetSecurityInfo](https://learn.microsoft.com/zh-cn/windows/desktop/api/aclapi/nf-aclapi-setsecurityinfo) |

## CloseServiceHandle函数的用法

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 关闭服务控制管理器或服务对象的句柄。 语法 C++复制  BOOL CloseServiceHandle(  [in] SC\_HANDLE hSCObject  ); 参数 [in] hSCObject  服务控制管理器对象的句柄或要关闭的服务对象的句柄。 服务控制管理器对象的句柄由 [OpenSCManager](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openscmanagera) 函数返回，服务对象的句柄由 [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea) 或 [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea) 函数返回。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 要获得更多的错误信息，请调用 GetLastError。  服务控制管理器可以设置以下错误代码。 其他错误代码可由服务控制管理器调用的注册表函数设置。  展开表   |  |  | | --- | --- | | **返回代码** | **说明** | | **ERROR\_INVALID\_HANDLE** | 指定的句柄无效。 |  注解 **CloseServiceHandle** 函数不会销毁句柄引用的服务控制管理器对象。 无法销毁服务控制管理器对象。 可以通过调用 [DeleteService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-deleteservice) 函数来销毁服务对象。 示例 有关示例，请参阅 [删除服务](https://learn.microsoft.com/zh-cn/windows/desktop/Services/deleting-a-service)。 要求  |  |  | | --- | --- | | **标头** | winsvc.h (包括 Windows.h) | | **Library** | Advapi32.lib | | **DLL** | Advapi32.dll |  另请参阅 [CreateService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-createservicea)  [DeleteService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-deleteservice)  [OpenSCManager](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openscmanagera)  [OpenService](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-openservicea)  [SCM 句柄](https://learn.microsoft.com/zh-cn/windows/desktop/Services/scm-handles)  [服务函数](https://learn.microsoft.com/zh-cn/windows/desktop/Services/service-functions) |

# 演练

## 我们想写能够自己安装自己的服务程序，安装服务使用servicedemo –i,删除服务使用servicedemo –r这样的格式来使用 ,我们的服务一旦创建会自动启动。

## 1.新建一个文件夹Lesson51-service-self-installer，然后把上一节的项目拷贝过来

|  |
| --- |
|  |

## 2.用vs2010打开这个项目，然后对他进行修改，首先需要给main函数添加命令行参数和修改返回类型

|  |
| --- |
|  |

## 3.然后我们需要根据命令行参数来决定操作，我们先声明2个函数

|  |
| --- |
|  |

## 4.然后需要在main函数里面根据命令行参数argv[1]来决定做什么操作

|  |
| --- |
|  |

## 5.然后需要添加这两个函数的实现，先做安装函数

|  |
| --- |
| //安装服务函数  BOOL CmdInstallService()  {  char szDir[1024]; //服务程序的路径  LPCTSTR lpBinPath;  SC\_HANDLE schMgr,schService;  //获取当前文件夹  GetCurrentDirectory(1024,szDir);  //获取应用程序名称  GetModuleFileName(NULL,szDir,sizeof(szDir));  lpBinPath = szDir;  schMgr = OpenSCManager(NULL,NULL,SC\_MANAGER\_ALL\_ACCESS);//打开服务管理器  if(schMgr == NULL)  {  printf("打开服务管理器失败,你可能没有操作权限\n");  return FALSE;  }  //打开服务管理器成功后，就可以创建服务,保存到服务句柄中。  schService = CreateService(schMgr,"TestInstallService","TestInstallService",SERVICE\_ALL\_ACCESS ,  SERVICE\_WIN32\_OWN\_PROCESS,SERVICE\_AUTO\_START,SERVICE\_ERROR\_NORMAL,lpBinPath,  NULL,NULL,NULL,NULL,NULL);  //检查服务是否创建成功  if(schService == NULL)  {  printf("创建服务器失败\n");  return FALSE;  }  //创建服务成功后需要把服务句柄关闭,因为这个服务创建成功后会自动运行，不需要句柄  CloseServiceHandle(schService);  return TRUE;  } |

## 6.然后我们来做CmdDeleteServeice函数

|  |
| --- |
| //删除服务的函数  void CmdDeleteService()  {  //删除服务之前，如果服务正在运行，需要先把服务停止，然后再删除  char name[100];  SC\_HANDLE sch,service;  SERVICE\_STATUS status;  BOOL isSuccess;  strcpy\_s(name,strlen("TestService"),"TestService");    if((sch = OpenSCManager(NULL,NULL,SC\_MANAGER\_CREATE\_SERVICE)) == NULL)  {  printf("打开服务管理器失败\n");  return;  }  service = OpenService(sch,name,SERVICE\_ALL\_ACCESS|DELETE);  if(service == NULL)  {  printf("打开%s服务失败\n",name);  return;  }  //查询服务状态  isSuccess = QueryServiceStatus(service,&status);  if(!isSuccess)  {  printf("查询服务状态失败\n");  return;  }  if(status.dwCurrentState !=SERVICE\_STOPPED)  {  //停止服务是用ControlService函数来实现  isSuccess = ControlService(service,SERVICE\_CONTROL\_STOP,&status);  if(!isSuccess)  {  printf("停止服务失败\n");  return;  }  }  //删除服务  isSuccess = DeleteService(service);  if(!isSuccess)  {  printf("删除服务失败\n");  }  else  {  printf("删除服务成功\n");  }  //删除完成后需要关闭句柄  CloseServiceHandle(service);  CloseServiceHandle(sch);  } |

## 7.重新生成项目，成功后如图

|  |
| --- |
|  |

## 8.把他复制粘贴到e:\service-self-installer文件夹里面

|  |
| --- |
|  |

## 9.先打开服务面板，确认当前没有安装这个服务

|  |
| --- |
|  |

### 然后打开一个cmd窗口，单位到这个目录下面，输入servicedemo –i 然后按回车安装服务

|  |
| --- |
|  |

### 注意：这个服务一旦安装，它会自动运行

|  |
| --- |
|  |

## 11，然后重新生成项目

|  |
| --- |
|  |

## 12. .把他复制粘贴到e:\service-self-installer文件夹里面

|  |
| --- |
|  |

### 13然后打开一个cmd窗口，单位到这个目录下面，输入SelfInService –i 然后按回车安装服务

|  |
| --- |
|  |

## 14，安装是安装成功了，而且也设置了自动运行，但是它并没有运行

|  |
| --- |
|  |

### 注意：这个自动指下一次windows启动的时候它会自动运行服务

## 15.把服务启动了，可以看到e盘下面生成了testlog.txt文件，双击打开，有日志记录说明服务工作正常

|  |
| --- |
|  |

## 16.然后我们可以把这个服务删除，注意删除服务只是把服务从服务面板里面移除，并没有删除exe文件

|  |
| --- |
|  |

## 扩展，可以编写一个类似以sc.exe的服务控制程序取名srvutil，用法srvutil –i servicename exefilepath

## 扩展2，可以编写一个FMC对话框程序来尝试在图形界面里面开启服务或者结束服务

# 这一节的学习到此为止，完整代码如下

## SelfinService.c

|  |
| --- |
| #include<Windows.h>  #include<stdio.h>  //定义一个睡眠时间常量  #define DURATION 10000  #define FILEPATH "e:\\testlog.txt"  SERVICE\_STATUS sStatus; //服务状态  SERVICE\_STATUS\_HANDLE ssHandle;  BOOL running = TRUE;//运行标记  void WINAPI ServiceMain(int argc,char\*\* argv);//声明服务入口函数  void WINAPI ServiceCtrlHandler(DWORD);//声明服务控制函数  int WriteLog(char \*str);//写日志函数  //声明服务安装和删除函数  BOOL CmdInstallService();  void CmdDeleteService();  int main(int argc,char\* argv[])  {  SERVICE\_TABLE\_ENTRY srvTable[2];//这个数组其实可以很大，也就是说可以有很多服务，但是最后一个必须是NULL，NULL，也就是说这里2其实代表只有一个服务，因为最后一个是NULL  if(argc == 2)  {  if(!strcmp(argv[1],"-i"))  {  CmdInstallService(); //-i 代表安装服务  }  else if(!strcmp(argv[1],"-r"))  {  CmdDeleteService();//-i 代表删除服务  }  }  srvTable[0].lpServiceName = "TestInstallService";//给服务取个名字  srvTable[0].lpServiceProc =(LPSERVICE\_MAIN\_FUNCTION)ServiceMain ;//指定服务的口函数入  srvTable[1].lpServiceName = NULL;  srvTable[1].lpServiceProc = NULL;  //注册服务入口函数  StartServiceCtrlDispatcher(srvTable);//注册成功后服务分派器就会给服务创建一个线程  return 0;  }  void WINAPI ServiceMain(int argc,char\*\* argv)  {  //内存操作需要的变量  MEMORYSTATUS memstatus;  char str[100];  int availMb = 0;//可用内存空间以MB为单位  //初始化服务状态属性  sStatus.dwServiceType = SERVICE\_WIN32;  sStatus.dwCurrentState = SERVICE\_START\_PENDING;//等待状态，因为服务还没有运行  sStatus.dwControlsAccepted = SERVICE\_ACCEPT\_SHUTDOWN|SERVICE\_ACCEPT\_STOP;//在等待状态下是允许关机和停止的  sStatus.dwWin32ExitCode = 0;//Win32退出码  sStatus.dwServiceSpecificExitCode = 0;//指定服务的退出码  sStatus.dwCheckPoint = 0;//一般用的很少设置为0即可  sStatus.dwWaitHint = 0; //一般用的很少设置为0即可  //注册服务控制函数  ssHandle = RegisterServiceCtrlHandler("TestInstallService",(LPHANDLER\_FUNCTION)ServiceCtrlHandler);  //如果注册失败，写到日志里面  if(ssHandle == 0)//注册失败返回0  {  WriteLog("Register Service Control Handler Failed!!");  }  //如果注册成功，也需要写日志  WriteLog("Register Service Control Handler Succeeded!!");  //注册成功后需要修改服务状态为正在运行  sStatus.dwCurrentState = SERVICE\_RUNNING;  //真正设置服务状态到服务控制句柄中，也就是相当于把状态公司服务控制函数  SetServiceStatus(ssHandle,&sStatus);  running = TRUE;    memset(str,0,100);  while(running)  {  //我们想在这个服务里面去检查电脑可用内存的大小  GlobalMemoryStatus(&memstatus);//获取内存当前状态  availMb = memstatus.dwAvailPhys/1024/1024;//可用物理内存  sprintf\_s(str,100,"可用物理内存为:%dMB\n",availMb);  WriteLog(str);  Sleep(DURATION);//让他睡眠一会，大概10秒钟  }  //循环结束也就是服务结束，此时也要写日志  WriteLog("服务停止了...");  }  //服务控制程序，控制服务的启动、暂停、停止、继续等等的操作  void WINAPI ServiceCtrlHandler(DWORD dwCtrl)  {  switch(dwCtrl)//注意这里的分支数目应该和上面dwControlsAccepted属性设置的操作数量一致  {  case SERVICE\_CONTROL\_STOP:  running= FALSE;  sStatus.dwCurrentState = SERVICE\_STOPPED;  break;  case SERVICE\_CONTROL\_SHUTDOWN:  running= FALSE;  break;  default:  break;  }  //报告服务状态和控制函数句柄  SetServiceStatus(ssHandle,&sStatus);  }  int WriteLog(char \*str)  {  FILE \*pfile;  fopen\_s(&pfile,FILEPATH,"a+");//打开文件，文件名以常量的形式定义好了  if(pfile == NULL)  {  printf("打开文件失败\n");  return -1;  }  fprintf\_s(pfile,"%s\n",str);  fclose(pfile);  return 0;  }  //定义服务安装和删除函数  //安装服务函数  BOOL CmdInstallService()  {  char szDir[1024]; //服务程序的路径  LPCTSTR lpBinPath;  SC\_HANDLE schMgr,schService;  //获取当前文件夹  GetCurrentDirectory(1024,szDir);  //获取应用程序名称  GetModuleFileName(NULL,szDir,sizeof(szDir));  lpBinPath = szDir;  schMgr = OpenSCManager(NULL,NULL,SC\_MANAGER\_ALL\_ACCESS);//打开服务管理器  if(schMgr == NULL)  {  printf("打开服务管理器失败,你可能没有操作权限\n");  return FALSE;  }  //打开服务管理器成功后，就可以创建服务,保存到服务句柄中。  schService = CreateService(schMgr,"TestInstallService","TestInstallService",SERVICE\_ALL\_ACCESS ,  SERVICE\_WIN32\_OWN\_PROCESS,SERVICE\_AUTO\_START,SERVICE\_ERROR\_NORMAL,lpBinPath,  NULL,NULL,NULL,NULL,NULL);  //检查服务是否创建成功  if(schService == NULL)  {  printf("创建服务器失败\n");  return FALSE;  }  //创建服务成功后需要把服务句柄关闭,因为这个服务创建成功后会自动运行，不需要句柄  CloseServiceHandle(schService);  return TRUE;  }  //删除服务的函数  void CmdDeleteService()  {  //删除服务之前，如果服务正在运行，需要先把服务停止，然后再删除  char name[100];  SC\_HANDLE sch,service;  SERVICE\_STATUS status;  BOOL isSuccess;  strcpy(name,"TestInstallService");    if((sch = OpenSCManager(NULL,NULL,SC\_MANAGER\_CREATE\_SERVICE)) == NULL)  {  printf("打开服务管理器失败\n");  return;  }  service = OpenService(sch,name,SERVICE\_ALL\_ACCESS|DELETE);  if(service == NULL)  {  printf("打开%s服务失败\n",name);  return;  }  //查询服务状态  isSuccess = QueryServiceStatus(service,&status);  if(!isSuccess)  {  printf("查询服务状态失败\n");  return;  }  if(status.dwCurrentState !=SERVICE\_STOPPED)  {  //停止服务是用ControlService函数来实现  isSuccess = ControlService(service,SERVICE\_CONTROL\_STOP,&status);  if(!isSuccess)  {  printf("停止服务失败\n");  return;  }  }  //删除服务  isSuccess = DeleteService(service);  if(!isSuccess)  {  printf("删除服务失败\n");  }  else  {  printf("删除服务成功\n");  }  //删除完成后需要关闭句柄  CloseServiceHandle(service);  CloseServiceHandle(sch);  } |