使用服务

# 本文内容

1. [服务计划任务](https://learn.microsoft.com/zh-cn/windows/win32/services/using-services#service-program-tasks)
2. [服务配置计划任务](https://learn.microsoft.com/zh-cn/windows/win32/services/using-services#service-configuration-program-tasks)
3. [服务控制程序任务](https://learn.microsoft.com/zh-cn/windows/win32/services/using-services#service-control-program-tasks)
4. [相关主题](https://learn.microsoft.com/zh-cn/windows/win32/services/using-services#related-topics)

若要创建和控制服务，请使用以下示例代码。

# 服务计划任务

## [编写服务程序的main函数](https://learn.microsoft.com/zh-cn/windows/win32/services/writing-a-service-program-s-main-function)

|  |
| --- |
| [服务程序的](https://learn.microsoft.com/zh-cn/windows/win32/services/service-programs)**main** 函数调用 [**StartServiceCtrlDispatcher**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-startservicectrldispatchera) 函数以连接到[服务控制管理器](https://learn.microsoft.com/zh-cn/windows/win32/services/service-control-manager) (SCM) 并启动控制调度程序线程。 调度程序线程循环，等待调度表中指定服务的传入控制请求。 当出现错误或进程中的所有服务已终止时，此线程将返回 。 当进程中的所有服务都终止时，SCM 会向调度程序线程发送控制请求，告知其退出。 然后，此线程从 **StartServiceCtrlDispatcher** 调用返回，进程可以终止。  此示例使用以下全局定义。  C++复制  #define SVCNAME TEXT("SvcName")  SERVICE\_STATUS gSvcStatus;  SERVICE\_STATUS\_HANDLE gSvcStatusHandle;  HANDLE ghSvcStopEvent = NULL;  以下示例可用作支持单个服务的服务程序的入口点。 如果服务程序支持多个服务，请将其他服务的名称添加到调度表，以便调度程序线程可以监视这些服务。  \_tmain函数是入口点。 SvcReportEvent 函数将信息性消息和错误写入事件日志。 有关编写 SvcMain 函数的信息，请参阅 [编写 ServiceMain 函数](https://learn.microsoft.com/zh-cn/windows/win32/services/writing-a-servicemain-function)。 有关 SvcInstall 函数的详细信息，请参阅 [安装服务](https://learn.microsoft.com/zh-cn/windows/win32/services/installing-a-service)。 有关编写 SvcCtrlHandler 函数的信息，请参阅 [编写控件处理程序函数](https://learn.microsoft.com/zh-cn/windows/win32/services/writing-a-control-handler-function)。 有关完整的示例服务（包括 SvcReportEvent 函数的源），请参阅 [Svc.cpp](https://learn.microsoft.com/zh-cn/windows/win32/services/svc-cpp)。  C++复制  //  // Purpose:  // Entry point for the process  //  // Parameters:  // None  //  // Return value:  // None, defaults to 0 (zero)  //  int \_\_cdecl \_tmain(int argc, TCHAR \*argv[])  {  // If command-line parameter is "install", install the service.  // Otherwise, the service is probably being started by the SCM.  if( lstrcmpi( argv[1], TEXT("install")) == 0 )  {  SvcInstall();  return;  }  // TO\_DO: Add any additional services for the process to this table.  SERVICE\_TABLE\_ENTRY DispatchTable[] =  {  { SVCNAME, (LPSERVICE\_MAIN\_FUNCTION) SvcMain },  { NULL, NULL }  };    // This call returns when the service has stopped.  // The process should simply terminate when the call returns.  if (!StartServiceCtrlDispatcher( DispatchTable ))  {  SvcReportEvent(TEXT("StartServiceCtrlDispatcher"));  }  }  下面是消息编译器生成的示例 Sample.h。 有关详细信息，请参阅 [Sample.mc](https://learn.microsoft.com/zh-cn/windows/win32/services/sample-mc)。  syntax复制  // The following are message definitions.  //  // Values are 32 bit values layed out as follows:  //  // 3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1  // 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0  // +---+-+-+-----------------------+-------------------------------+  // |Sev|C|R| Facility | Code |  // +---+-+-+-----------------------+-------------------------------+  //  // where  //  // Sev - is the severity code  //  // 00 - Success  // 01 - Informational  // 10 - Warning  // 11 - Error  //  // C - is the Customer code flag  //  // R - is a reserved bit  //  // Facility - is the facility code  //  // Code - is the facility's status code  //  //  // Define the facility codes  //  #define FACILITY\_SYSTEM 0x0  #define FACILITY\_STUBS 0x3  #define FACILITY\_RUNTIME 0x2  #define FACILITY\_IO\_ERROR\_CODE 0x4  //  // Define the severity codes  //  #define STATUS\_SEVERITY\_WARNING 0x2  #define STATUS\_SEVERITY\_SUCCESS 0x0  #define STATUS\_SEVERITY\_INFORMATIONAL 0x1  #define STATUS\_SEVERITY\_ERROR 0x3  //  // MessageId: SVC\_ERROR  //  // MessageText:  //  // An error has occurred (%2).  //  //  #define SVC\_ERROR ((DWORD)0xC0020001L) |

## [编写 ServiceMain 函数](https://learn.microsoft.com/zh-cn/windows/win32/services/writing-a-servicemain-function)

|  |
| --- |
| 以下示例中的 SvcMain 函数是示例服务的 [**ServiceMain**](https://learn.microsoft.com/zh-cn/windows/win32/api/winsvc/nc-winsvc-lpservice_main_functiona) 函数。 SvcMain 可以按照控制台应用程序的 **main** 函数的方式访问服务的命令行参数。 第一个参数包含在第二个参数中传递给服务的参数数。 始终至少有一个参数。 第二个参数是指向字符串指针数组的指针。 数组中的第一项始终是服务名称。  SvcMain 函数首先调用 [**RegisterServiceCtrlHandler**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-registerservicectrlhandlera) 函数，以将 SvcCtrlHandler 函数注册为服务的 [**Handler**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nc-winsvc-lphandler_function) 函数并开始初始化。 **RegisterServiceCtrlHandler** 应为 [**ServiceMain**](https://learn.microsoft.com/zh-cn/windows/win32/api/winsvc/nc-winsvc-lpservice_main_functiona) 中的第一个非失败函数，以便服务可以使用此函数返回的状态句柄在发生错误时调用具有SERVICE\_STOPPED状态的 [**SetServiceStatus**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-setservicestatus) 。  接下来，SvcMain 函数调用 ReportSvcStatus 函数以指示其初始状态为SERVICE\_START\_PENDING。 当服务处于此状态时，不会接受任何控件。 为了简化服务的逻辑，建议服务在执行初始化时不要接受任何控件。  最后，SvcMain 函数调用 SvcInit 函数来执行特定于服务的初始化，并开始服务要执行的工作。  示例初始化函数 SvcInit 是一个非常简单的示例;它不会执行更复杂的初始化任务，例如创建其他线程。 它创建一个事件，服务控制处理程序可以发出信号来指示服务应停止，然后调用 ReportSvcStatus 以指示服务已进入SERVICE\_RUNNING状态。 此时，服务已完成其初始化，并已准备好接受控制。 为获得最佳系统性能，应用程序应在 25-100 毫秒内进入运行状态。  由于此示例服务未完成任何实际任务，SvcInit 只需通过调用 [**WaitForSingleObject**](https://learn.microsoft.com/zh-cn/windows/desktop/api/synchapi/nf-synchapi-waitforsingleobject) 函数来等待服务停止事件发出信号，调用 ReportSvcStatus 以指示服务已进入SERVICE\_STOPPED状态，并返回 。 (请注意，函数返回而不是调用 [**ExitThread**](https://learn.microsoft.com/zh-cn/windows/desktop/api/processthreadsapi/nf-processthreadsapi-exitthread) 函数非常重要，因为返回 允许清理为 arguments.) 可以使用 [**RegisterWaitForSingleObject**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winbase/nf-winbase-registerwaitforsingleobject) 函数而不是 **WaitForSingleObject** 函数来执行其他清理任务。 运行 [**ServiceMain**](https://learn.microsoft.com/zh-cn/windows/win32/api/winsvc/nc-winsvc-lpservice_main_functiona) 函数的线程终止，但服务本身继续运行。 当服务控制处理程序发出事件信号时，线程池中的线程将执行回调以执行其他清理，包括将状态设置为SERVICE\_STOPPED。  请注意，此示例使用 SvcReportEvent 将错误事件写入事件日志。 有关 SvcReportEvent 的源代码，请参阅 [Svc.cpp](https://learn.microsoft.com/zh-cn/windows/win32/services/svc-cpp)。 有关控件处理程序函数的示例，请参阅 [编写控件处理程序函数](https://learn.microsoft.com/zh-cn/windows/win32/services/writing-a-control-handler-function)。  此示例使用以下全局定义。  C++复制  #define SVCNAME TEXT("SvcName")  SERVICE\_STATUS gSvcStatus;  SERVICE\_STATUS\_HANDLE gSvcStatusHandle;  HANDLE ghSvcStopEvent = NULL;  以下示例片段取自完整的服务示例。  C++复制  //  // Purpose:  // Entry point for the service  //  // Parameters:  // dwArgc - Number of arguments in the lpszArgv array  // lpszArgv - Array of strings. The first string is the name of  // the service and subsequent strings are passed by the process  // that called the StartService function to start the service.  //  // Return value:  // None.  //  VOID WINAPI SvcMain( DWORD dwArgc, LPTSTR \*lpszArgv )  {  // Register the handler function for the service  gSvcStatusHandle = RegisterServiceCtrlHandler(  SVCNAME,  SvcCtrlHandler);  if( !gSvcStatusHandle )  {  SvcReportEvent(TEXT("RegisterServiceCtrlHandler"));  return;  }  // These SERVICE\_STATUS members remain as set here  gSvcStatus.dwServiceType = SERVICE\_WIN32\_OWN\_PROCESS;  gSvcStatus.dwServiceSpecificExitCode = 0;  // Report initial status to the SCM  ReportSvcStatus( SERVICE\_START\_PENDING, NO\_ERROR, 3000 );  // Perform service-specific initialization and work.  SvcInit( dwArgc, lpszArgv );  }  //  // Purpose:  // The service code  //  // Parameters:  // dwArgc - Number of arguments in the lpszArgv array  // lpszArgv - Array of strings. The first string is the name of  // the service and subsequent strings are passed by the process  // that called the StartService function to start the service.  //  // Return value:  // None  //  VOID SvcInit( DWORD dwArgc, LPTSTR \*lpszArgv)  {  // TO\_DO: Declare and set any required variables.  // Be sure to periodically call ReportSvcStatus() with  // SERVICE\_START\_PENDING. If initialization fails, call  // ReportSvcStatus with SERVICE\_STOPPED.  // Create an event. The control handler function, SvcCtrlHandler,  // signals this event when it receives the stop control code.  ghSvcStopEvent = CreateEvent(  NULL, // default security attributes  TRUE, // manual reset event  FALSE, // not signaled  NULL); // no name  if ( ghSvcStopEvent == NULL)  {  ReportSvcStatus( SERVICE\_STOPPED, NO\_ERROR, 0 );  return;  }  // Report running status when initialization is complete.  ReportSvcStatus( SERVICE\_RUNNING, NO\_ERROR, 0 );  // TO\_DO: Perform work until service stops.  while(1)  {  // Check whether to stop the service.  WaitForSingleObject(ghSvcStopEvent, INFINITE);  ReportSvcStatus( SERVICE\_STOPPED, NO\_ERROR, 0 );  return;  }  }  //  // Purpose:  // Sets the current service status and reports it to the SCM.  //  // Parameters:  // dwCurrentState - The current state (see SERVICE\_STATUS)  // dwWin32ExitCode - The system error code  // dwWaitHint - Estimated time for pending operation,  // in milliseconds  //  // Return value:  // None  //  VOID ReportSvcStatus( DWORD dwCurrentState,  DWORD dwWin32ExitCode,  DWORD dwWaitHint)  {  static DWORD dwCheckPoint = 1;  // Fill in the SERVICE\_STATUS structure.  gSvcStatus.dwCurrentState = dwCurrentState;  gSvcStatus.dwWin32ExitCode = dwWin32ExitCode;  gSvcStatus.dwWaitHint = dwWaitHint;  if (dwCurrentState == SERVICE\_START\_PENDING)  gSvcStatus.dwControlsAccepted = 0;  else gSvcStatus.dwControlsAccepted = SERVICE\_ACCEPT\_STOP;  if ( (dwCurrentState == SERVICE\_RUNNING) ||  (dwCurrentState == SERVICE\_STOPPED) )  gSvcStatus.dwCheckPoint = 0;  else gSvcStatus.dwCheckPoint = dwCheckPoint++;  // Report the status of the service to the SCM.  SetServiceStatus( gSvcStatusHandle, &gSvcStatus );  } |

## [编写控件处理程序函数](https://learn.microsoft.com/zh-cn/windows/win32/services/writing-a-control-handler-function)

|  |
| --- |
| 当调度程序线程调用 [**Handler**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nc-winsvc-lphandler_function) 函数时，它会处理 *在 Opcode* 参数中传递的控制代码，然后调用 ReportSvcStatus 函数来更新服务状态。 当 [**Handler**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nc-winsvc-lphandler_function) 函数收到控制代码时，仅当处理控制代码导致服务状态更改时，它才应报告服务状态。 如果服务不对控件执行操作，则不应向服务控制管理器报告状态。 有关 ReportSvcStatus 的源代码，请参阅 [编写 ServiceMain 函数](https://learn.microsoft.com/zh-cn/windows/win32/services/writing-a-servicemain-function)。  在下面的示例中，SvcCtrlHandler 函数是 [**Handler**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nc-winsvc-lphandler_function) 函数的一个示例。 请注意，ghSvcStopEvent 变量是一个全局变量，应初始化并使用，如 [编写 ServiceMain 函数](https://learn.microsoft.com/zh-cn/windows/win32/services/writing-a-servicemain-function)中所述。  C++复制  //  // Purpose:  // Called by SCM whenever a control code is sent to the service  // using the ControlService function.  //  // Parameters:  // dwCtrl - control code  //  // Return value:  // None  //  VOID WINAPI SvcCtrlHandler( DWORD dwCtrl )  {  // Handle the requested control code.  switch(dwCtrl)  {  case SERVICE\_CONTROL\_STOP:  ReportSvcStatus(SERVICE\_STOP\_PENDING, NO\_ERROR, 0);  // Signal the service to stop.  SetEvent(ghSvcStopEvent);  ReportSvcStatus(gSvcStatus.dwCurrentState, NO\_ERROR, 0);    return;    case SERVICE\_CONTROL\_INTERROGATE:  break;    default:  break;  }    } |

# 服务配置计划任务

## [安装服务](https://learn.microsoft.com/zh-cn/windows/win32/services/installing-a-service)

|  |
| --- |
| [服务配置程序](https://learn.microsoft.com/zh-cn/windows/win32/services/service-configuration-programs)使用 [**CreateService**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-createservicea) 函数在 SCM 数据库中安装服务。  以下示例中的 SvcInstall 函数演示如何从服务程序本身安装服务。 有关完整示例，请参阅 [Svc.cpp](https://learn.microsoft.com/zh-cn/windows/win32/services/svc-cpp)。  C++复制  //  // Purpose:  // Installs a service in the SCM database  //  // Parameters:  // None  //  // Return value:  // None  //  VOID SvcInstall()  {  SC\_HANDLE schSCManager;  SC\_HANDLE schService;  TCHAR szUnquotedPath[MAX\_PATH];  if( !GetModuleFileName( NULL, szUnquotedPath, MAX\_PATH ) )  {  printf("Cannot install service (%d)\n", GetLastError());  return;  }  // In case the path contains a space, it must be quoted so that  // it is correctly interpreted. For example,  // "d:\my share\myservice.exe" should be specified as  // ""d:\my share\myservice.exe"".  TCHAR szPath[MAX\_PATH];  StringCbPrintf(szPath, MAX\_PATH, TEXT("\"%s\""), szUnquotedPath);  // Get a handle to the SCM database.    schSCManager = OpenSCManager(  NULL, // local computer  NULL, // ServicesActive database  SC\_MANAGER\_ALL\_ACCESS); // full access rights    if (NULL == schSCManager)  {  printf("OpenSCManager failed (%d)\n", GetLastError());  return;  }  // Create the service  schService = CreateService(  schSCManager, // SCM database  SVCNAME, // name of service  SVCNAME, // service name to display  SERVICE\_ALL\_ACCESS, // desired access  SERVICE\_WIN32\_OWN\_PROCESS, // service type  SERVICE\_DEMAND\_START, // start type  SERVICE\_ERROR\_NORMAL, // error control type  szPath, // path to service's binary  NULL, // no load ordering group  NULL, // no tag identifier  NULL, // no dependencies  NULL, // LocalSystem account  NULL); // no password    if (schService == NULL)  {  printf("CreateService failed (%d)\n", GetLastError());  CloseServiceHandle(schSCManager);  return;  }  else printf("Service installed successfully\n");  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  } |

## [删除服务](https://learn.microsoft.com/zh-cn/windows/win32/services/deleting-a-service)

|  |
| --- |
| [服务配置程序](https://learn.microsoft.com/zh-cn/windows/win32/services/service-configuration-programs)使用 [**OpenService**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-openservicea) 函数获取已安装服务对象的句柄。 然后，程序可以使用 [**DeleteService**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-deleteservice) 函数中的服务对象句柄从 SCM 数据库中删除服务。  以下示例中的 DoDeleteSvc 函数演示如何从 SCM 数据库中删除服务。 szSvcName 变量是一个全局变量，其中包含要删除的服务的名称。 有关设置此变量的完整示例，请参阅 [SvcConfig.cpp](https://learn.microsoft.com/zh-cn/windows/win32/services/svcconfig-cpp)。  C++复制  //  // Purpose:  // Deletes a service from the SCM database  //  // Parameters:  // None  //  // Return value:  // None  //  VOID \_\_stdcall DoDeleteSvc()  {  SC\_HANDLE schSCManager;  SC\_HANDLE schService;  SERVICE\_STATUS ssStatus;  // Get a handle to the SCM database.    schSCManager = OpenSCManager(  NULL, // local computer  NULL, // ServicesActive database  SC\_MANAGER\_ALL\_ACCESS); // full access rights    if (NULL == schSCManager)  {  printf("OpenSCManager failed (%d)\n", GetLastError());  return;  }  // Get a handle to the service.  schService = OpenService(  schSCManager, // SCM database  szSvcName, // name of service  DELETE); // need delete access    if (schService == NULL)  {  printf("OpenService failed (%d)\n", GetLastError());  CloseServiceHandle(schSCManager);  return;  }  // Delete the service.    if (! DeleteService(schService) )  {  printf("DeleteService failed (%d)\n", GetLastError());  }  else printf("Service deleted successfully\n");    CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  } |

## [更改服务的配置](https://learn.microsoft.com/zh-cn/windows/win32/services/changing-a-service-configuration)

|  |
| --- |
| [服务配置程序](https://learn.microsoft.com/zh-cn/windows/win32/services/service-configuration-programs)使用 [**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) 函数更改已安装服务的配置参数。 程序打开服务对象的句柄，修改其配置，然后关闭服务对象句柄。  在以下示例中，DoDisableSvc 函数使用 [**ChangeServiceConfig**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-changeserviceconfiga) 将服务启动类型更改为“Disabled”，DoEnableSvc 函数使用 **ChangeServiceConfig** 将服务启动类型更改为“Enabled”，DoUpdateSvcDesc 函数使用 [**ChangeServiceConfig2**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-changeserviceconfig2a) 将服务说明设置为“这是测试说明”。 szSvcName 变量是包含服务名称的全局变量。 有关设置此变量的完整示例，请参阅 [SvcConfig.cpp](https://learn.microsoft.com/zh-cn/windows/win32/services/svcconfig-cpp)。  C++复制  //  // Purpose:  // Disables the service.  //  // Parameters:  // None  //  // Return value:  // None  //  VOID \_\_stdcall DoDisableSvc()  {  SC\_HANDLE schSCManager;  SC\_HANDLE schService;  // Get a handle to the SCM database.    schSCManager = OpenSCManager(  NULL, // local computer  NULL, // ServicesActive database  SC\_MANAGER\_ALL\_ACCESS); // full access rights    if (NULL == schSCManager)  {  printf("OpenSCManager failed (%d)\n", GetLastError());  return;  }  // Get a handle to the service.  schService = OpenService(  schSCManager, // SCM database  szSvcName, // name of service  SERVICE\_CHANGE\_CONFIG); // need change config access    if (schService == NULL)  {  printf("OpenService failed (%d)\n", GetLastError());  CloseServiceHandle(schSCManager);  return;  }  // Change the service start type.  if (! ChangeServiceConfig(  schService, // handle of service  SERVICE\_NO\_CHANGE, // service type: no change  SERVICE\_DISABLED, // service start type  SERVICE\_NO\_CHANGE, // error control: no change  NULL, // binary path: no change  NULL, // load order group: no change  NULL, // tag ID: no change  NULL, // dependencies: no change  NULL, // account name: no change  NULL, // password: no change  NULL) ) // display name: no change  {  printf("ChangeServiceConfig failed (%d)\n", GetLastError());  }  else printf("Service disabled successfully.\n");  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  }  //  // Purpose:  // Enables the service.  //  // Parameters:  // None  //  // Return value:  // None  //  VOID \_\_stdcall DoEnableSvc()  {  SC\_HANDLE schSCManager;  SC\_HANDLE schService;  // Get a handle to the SCM database.    schSCManager = OpenSCManager(  NULL, // local computer  NULL, // ServicesActive database  SC\_MANAGER\_ALL\_ACCESS); // full access rights    if (NULL == schSCManager)  {  printf("OpenSCManager failed (%d)\n", GetLastError());  return;  }  // Get a handle to the service.  schService = OpenService(  schSCManager, // SCM database  szSvcName, // name of service  SERVICE\_CHANGE\_CONFIG); // need change config access    if (schService == NULL)  {  printf("OpenService failed (%d)\n", GetLastError());  CloseServiceHandle(schSCManager);  return;  }  // Change the service start type.  if (! ChangeServiceConfig(  schService, // handle of service  SERVICE\_NO\_CHANGE, // service type: no change  SERVICE\_DEMAND\_START, // service start type  SERVICE\_NO\_CHANGE, // error control: no change  NULL, // binary path: no change  NULL, // load order group: no change  NULL, // tag ID: no change  NULL, // dependencies: no change  NULL, // account name: no change  NULL, // password: no change  NULL) ) // display name: no change  {  printf("ChangeServiceConfig failed (%d)\n", GetLastError());  }  else printf("Service enabled successfully.\n");  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  }  //  // Purpose:  // Updates the service description to "This is a test description".  //  // Parameters:  // None  //  // Return value:  // None  //  VOID \_\_stdcall DoUpdateSvcDesc()  {  SC\_HANDLE schSCManager;  SC\_HANDLE schService;  SERVICE\_DESCRIPTION sd;  LPTSTR szDesc = TEXT("This is a test description");  // Get a handle to the SCM database.    schSCManager = OpenSCManager(  NULL, // local computer  NULL, // ServicesActive database  SC\_MANAGER\_ALL\_ACCESS); // full access rights    if (NULL == schSCManager)  {  printf("OpenSCManager failed (%d)\n", GetLastError());  return;  }  // Get a handle to the service.  schService = OpenService(  schSCManager, // SCM database  szSvcName, // name of service  SERVICE\_CHANGE\_CONFIG); // need change config access    if (schService == NULL)  {  printf("OpenService failed (%d)\n", GetLastError());  CloseServiceHandle(schSCManager);  return;  }  // Change the service description.  sd.lpDescription = szDesc;  if( !ChangeServiceConfig2(  schService, // handle to service  SERVICE\_CONFIG\_DESCRIPTION, // change: description  &sd) ) // new description  {  printf("ChangeServiceConfig2 failed\n");  }  else printf("Service description updated successfully.\n");  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  } |

## [查询服务配置](https://learn.microsoft.com/zh-cn/windows/win32/services/querying-a-service-s-configuration)

|  |
| --- |
| [服务配置程序](https://learn.microsoft.com/zh-cn/windows/win32/services/service-configuration-programs)使用 [**OpenService**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-openservicea) 函数获取句柄，SERVICE\_QUERY\_CONFIG访问已安装的服务对象。 然后，程序可以使用 [**QueryServiceConfig**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-queryserviceconfiga) 和 [**QueryServiceConfig2**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-queryserviceconfig2a) 函数中的服务对象句柄来检索服务的当前配置。  在以下示例中，DoQuerySvc 函数使用 [**QueryServiceConfig**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-queryserviceconfiga) 和 [**QueryServiceConfig2**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-queryserviceconfig2a) 检索配置信息，然后将所选信息写入控制台。 szSvcName 变量是包含服务名称的全局变量。 有关设置此变量的完整示例，请参阅 [SvcConfig.cpp](https://learn.microsoft.com/zh-cn/windows/win32/services/svcconfig-cpp)。  C++复制  //  // Purpose:  // Retrieves and displays the current service configuration.  //  // Parameters:  // None  //  // Return value:  // None  //  VOID \_\_stdcall DoQuerySvc()  {  SC\_HANDLE schSCManager;  SC\_HANDLE schService;  LPQUERY\_SERVICE\_CONFIG lpsc;  LPSERVICE\_DESCRIPTION lpsd;  DWORD dwBytesNeeded, cbBufSize, dwError;  // Get a handle to the SCM database.    schSCManager = OpenSCManager(  NULL, // local computer  NULL, // ServicesActive database  SC\_MANAGER\_ALL\_ACCESS); // full access rights    if (NULL == schSCManager)  {  printf("OpenSCManager failed (%d)\n", GetLastError());  return;  }  // Get a handle to the service.  schService = OpenService(  schSCManager, // SCM database  szSvcName, // name of service  SERVICE\_QUERY\_CONFIG); // need query config access    if (schService == NULL)  {  printf("OpenService failed (%d)\n", GetLastError());  CloseServiceHandle(schSCManager);  return;  }  // Get the configuration information.    if( !QueryServiceConfig(  schService,  NULL,  0,  &dwBytesNeeded))  {  dwError = GetLastError();  if( ERROR\_INSUFFICIENT\_BUFFER == dwError )  {  cbBufSize = dwBytesNeeded;  lpsc = (LPQUERY\_SERVICE\_CONFIG) LocalAlloc(LMEM\_FIXED, cbBufSize);  }  else  {  printf("QueryServiceConfig failed (%d)", dwError);  goto cleanup;  }  }    if( !QueryServiceConfig(  schService,  lpsc,  cbBufSize,  &dwBytesNeeded) )  {  printf("QueryServiceConfig failed (%d)", GetLastError());  goto cleanup;  }  if( !QueryServiceConfig2(  schService,  SERVICE\_CONFIG\_DESCRIPTION,  NULL,  0,  &dwBytesNeeded))  {  dwError = GetLastError();  if( ERROR\_INSUFFICIENT\_BUFFER == dwError )  {  cbBufSize = dwBytesNeeded;  lpsd = (LPSERVICE\_DESCRIPTION) LocalAlloc(LMEM\_FIXED, cbBufSize);  }  else  {  printf("QueryServiceConfig2 failed (%d)", dwError);  goto cleanup;  }  }    if (! QueryServiceConfig2(  schService,  SERVICE\_CONFIG\_DESCRIPTION,  (LPBYTE) lpsd,  cbBufSize,  &dwBytesNeeded) )  {  printf("QueryServiceConfig2 failed (%d)", GetLastError());  goto cleanup;  }    // Print the configuration information.    \_tprintf(TEXT("%s configuration: \n"), szSvcName);  \_tprintf(TEXT(" Type: 0x%x\n"), lpsc->dwServiceType);  \_tprintf(TEXT(" Start Type: 0x%x\n"), lpsc->dwStartType);  \_tprintf(TEXT(" Error Control: 0x%x\n"), lpsc->dwErrorControl);  \_tprintf(TEXT(" Binary path: %s\n"), lpsc->lpBinaryPathName);  \_tprintf(TEXT(" Account: %s\n"), lpsc->lpServiceStartName);  if (lpsd->lpDescription != NULL && lstrcmp(lpsd->lpDescription, TEXT("")) != 0)  \_tprintf(TEXT(" Description: %s\n"), lpsd->lpDescription);  if (lpsc->lpLoadOrderGroup != NULL && lstrcmp(lpsc->lpLoadOrderGroup, TEXT("")) != 0)  \_tprintf(TEXT(" Load order group: %s\n"), lpsc->lpLoadOrderGroup);  if (lpsc->dwTagId != 0)  \_tprintf(TEXT(" Tag ID: %d\n"), lpsc->dwTagId);  if (lpsc->lpDependencies != NULL && lstrcmp(lpsc->lpDependencies, TEXT("")) != 0)  \_tprintf(TEXT(" Dependencies: %s\n"), lpsc->lpDependencies);    LocalFree(lpsc);  LocalFree(lpsd);  cleanup:  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  } |

# 服务控制程序任务

## [启动服务](https://learn.microsoft.com/zh-cn/windows/win32/services/starting-a-service)

|  |
| --- |
| 若要启动服务， [服务控制程序](https://learn.microsoft.com/zh-cn/windows/win32/services/service-control-programs) 会打开已安装数据库的句柄，然后在对 [**StartService**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-startservicea) 函数的调用中指定句柄。 启动服务后，程序使用 [**QueryServiceStatusEx**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-queryservicestatusex) 函数返回[**的 SERVICE\_STATUS\_PROCESS**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/ns-winsvc-service_status_process) 结构的成员来跟踪服务的进度。  以下示例中的 DoStartSvc 函数演示如何启动服务。 szSvcName 变量是一个全局变量，其中包含要启动的服务的名称。 有关设置此变量的完整示例，请参阅 [SvcControl.cpp](https://learn.microsoft.com/zh-cn/windows/win32/services/svccontrol-cpp)。  C++复制  //  // Purpose:  // Starts the service if possible.  //  // Parameters:  // None  //  // Return value:  // None  //  VOID \_\_stdcall DoStartSvc()  {  SERVICE\_STATUS\_PROCESS ssStatus;  DWORD dwOldCheckPoint;  DWORD dwStartTickCount;  DWORD dwWaitTime;  DWORD dwBytesNeeded;  // Get a handle to the SCM database.    schSCManager = OpenSCManager(  NULL, // local computer  NULL, // servicesActive database  SC\_MANAGER\_ALL\_ACCESS); // full access rights    if (NULL == schSCManager)  {  printf("OpenSCManager failed (%d)\n", GetLastError());  return;  }  // Get a handle to the service.  schService = OpenService(  schSCManager, // SCM database  szSvcName, // name of service  SERVICE\_ALL\_ACCESS); // full access    if (schService == NULL)  {  printf("OpenService failed (%d)\n", GetLastError());  CloseServiceHandle(schSCManager);  return;  }  // Check the status in case the service is not stopped.  if (!QueryServiceStatusEx(  schService, // handle to service  SC\_STATUS\_PROCESS\_INFO, // information level  (LPBYTE) &ssStatus, // address of structure  sizeof(SERVICE\_STATUS\_PROCESS), // size of structure  &dwBytesNeeded ) ) // size needed if buffer is too small  {  printf("QueryServiceStatusEx failed (%d)\n", GetLastError());  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  return;  }  // Check if the service is already running. It would be possible  // to stop the service here, but for simplicity this example just returns.  if(ssStatus.dwCurrentState != SERVICE\_STOPPED && ssStatus.dwCurrentState != SERVICE\_STOP\_PENDING)  {  printf("Cannot start the service because it is already running\n");  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  return;  }  // Save the tick count and initial checkpoint.  dwStartTickCount = GetTickCount();  dwOldCheckPoint = ssStatus.dwCheckPoint;  // Wait for the service to stop before attempting to start it.  while (ssStatus.dwCurrentState == SERVICE\_STOP\_PENDING)  {  // Do not wait longer than the wait hint. A good interval is  // one-tenth of the wait hint but not less than 1 second  // and not more than 10 seconds.    dwWaitTime = ssStatus.dwWaitHint / 10;  if( dwWaitTime < 1000 )  dwWaitTime = 1000;  else if ( dwWaitTime > 10000 )  dwWaitTime = 10000;  Sleep( dwWaitTime );  // Check the status until the service is no longer stop pending.    if (!QueryServiceStatusEx(  schService, // handle to service  SC\_STATUS\_PROCESS\_INFO, // information level  (LPBYTE) &ssStatus, // address of structure  sizeof(SERVICE\_STATUS\_PROCESS), // size of structure  &dwBytesNeeded ) ) // size needed if buffer is too small  {  printf("QueryServiceStatusEx failed (%d)\n", GetLastError());  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  return;  }  if ( ssStatus.dwCheckPoint > dwOldCheckPoint )  {  // Continue to wait and check.  dwStartTickCount = GetTickCount();  dwOldCheckPoint = ssStatus.dwCheckPoint;  }  else  {  if(GetTickCount()-dwStartTickCount > ssStatus.dwWaitHint)  {  printf("Timeout waiting for service to stop\n");  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  return;  }  }  }  // Attempt to start the service.  if (!StartService(  schService, // handle to service  0, // number of arguments  NULL) ) // no arguments  {  printf("StartService failed (%d)\n", GetLastError());  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  return;  }  else printf("Service start pending...\n");  // Check the status until the service is no longer start pending.    if (!QueryServiceStatusEx(  schService, // handle to service  SC\_STATUS\_PROCESS\_INFO, // info level  (LPBYTE) &ssStatus, // address of structure  sizeof(SERVICE\_STATUS\_PROCESS), // size of structure  &dwBytesNeeded ) ) // if buffer too small  {  printf("QueryServiceStatusEx failed (%d)\n", GetLastError());  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  return;  }    // Save the tick count and initial checkpoint.  dwStartTickCount = GetTickCount();  dwOldCheckPoint = ssStatus.dwCheckPoint;  while (ssStatus.dwCurrentState == SERVICE\_START\_PENDING)  {  // Do not wait longer than the wait hint. A good interval is  // one-tenth the wait hint, but no less than 1 second and no  // more than 10 seconds.    dwWaitTime = ssStatus.dwWaitHint / 10;  if( dwWaitTime < 1000 )  dwWaitTime = 1000;  else if ( dwWaitTime > 10000 )  dwWaitTime = 10000;  Sleep( dwWaitTime );  // Check the status again.    if (!QueryServiceStatusEx(  schService, // handle to service  SC\_STATUS\_PROCESS\_INFO, // info level  (LPBYTE) &ssStatus, // address of structure  sizeof(SERVICE\_STATUS\_PROCESS), // size of structure  &dwBytesNeeded ) ) // if buffer too small  {  printf("QueryServiceStatusEx failed (%d)\n", GetLastError());  break;  }    if ( ssStatus.dwCheckPoint > dwOldCheckPoint )  {  // Continue to wait and check.  dwStartTickCount = GetTickCount();  dwOldCheckPoint = ssStatus.dwCheckPoint;  }  else  {  if(GetTickCount()-dwStartTickCount > ssStatus.dwWaitHint)  {  // No progress made within the wait hint.  break;  }  }  }  // Determine whether the service is running.  if (ssStatus.dwCurrentState == SERVICE\_RUNNING)  {  printf("Service started successfully.\n");  }  else  {  printf("Service not started. \n");  printf(" Current State: %d\n", ssStatus.dwCurrentState);  printf(" Exit Code: %d\n", ssStatus.dwWin32ExitCode);  printf(" Check Point: %d\n", ssStatus.dwCheckPoint);  printf(" Wait Hint: %d\n", ssStatus.dwWaitHint);  }  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  } |

## [停止服务](https://learn.microsoft.com/zh-cn/windows/win32/services/stopping-a-service)

|  |
| --- |
| [服务控制程序](https://learn.microsoft.com/zh-cn/windows/win32/services/service-control-programs)可以通过使用 [**ControlService**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Winsvc/nf-winsvc-controlservice) 函数发送SERVICE\_CONTROL\_STOP请求来停止服务。 如果 [服务控制管理器](https://learn.microsoft.com/zh-cn/windows/win32/services/service-control-manager) (SCM) 收到服务SERVICE\_CONTROL\_STOP请求，它会通过将请求转发到服务的 [**ServiceMain**](https://learn.microsoft.com/zh-cn/windows/win32/api/winsvc/nc-winsvc-lpservice_main_functiona) 函数来指示服务停止。 但是，如果 SCM 确定其他正在运行的服务依赖于指定的服务，则不会转发停止请求。 而是返回ERROR\_DEPENDENT\_SERVICES\_RUNNING。 因此，若要以编程方式停止此类服务，必须先枚举并停止其依赖服务。  以下示例中的 DoStopSvc 函数演示如何停止服务和任何依赖服务。 szSvcName 变量是一个全局变量，其中包含要停止的服务的名称。 有关设置此变量的完整示例，请参阅 [SvcControl.cpp](https://learn.microsoft.com/zh-cn/windows/win32/services/svccontrol-cpp)。  C++复制  //  // Purpose:  // Stops the service.  //  // Parameters:  // None  //  // Return value:  // None  //  VOID \_\_stdcall DoStopSvc()  {  SERVICE\_STATUS\_PROCESS ssp;  DWORD dwStartTime = GetTickCount();  DWORD dwBytesNeeded;  DWORD dwTimeout = 30000; // 30-second time-out  DWORD dwWaitTime;  // Get a handle to the SCM database.    schSCManager = OpenSCManager(  NULL, // local computer  NULL, // ServicesActive database  SC\_MANAGER\_ALL\_ACCESS); // full access rights    if (NULL == schSCManager)  {  printf("OpenSCManager failed (%d)\n", GetLastError());  return;  }  // Get a handle to the service.  schService = OpenService(  schSCManager, // SCM database  szSvcName, // name of service  SERVICE\_STOP |  SERVICE\_QUERY\_STATUS |  SERVICE\_ENUMERATE\_DEPENDENTS);    if (schService == NULL)  {  printf("OpenService failed (%d)\n", GetLastError());  CloseServiceHandle(schSCManager);  return;  }  // Make sure the service is not already stopped.  if ( !QueryServiceStatusEx(  schService,  SC\_STATUS\_PROCESS\_INFO,  (LPBYTE)&ssp,  sizeof(SERVICE\_STATUS\_PROCESS),  &dwBytesNeeded ) )  {  printf("QueryServiceStatusEx failed (%d)\n", GetLastError());  goto stop\_cleanup;  }  if ( ssp.dwCurrentState == SERVICE\_STOPPED )  {  printf("Service is already stopped.\n");  goto stop\_cleanup;  }  // If a stop is pending, wait for it.  while ( ssp.dwCurrentState == SERVICE\_STOP\_PENDING )  {  printf("Service stop pending...\n");  // Do not wait longer than the wait hint. A good interval is  // one-tenth of the wait hint but not less than 1 second  // and not more than 10 seconds.    dwWaitTime = ssp.dwWaitHint / 10;  if( dwWaitTime < 1000 )  dwWaitTime = 1000;  else if ( dwWaitTime > 10000 )  dwWaitTime = 10000;  Sleep( dwWaitTime );  if ( !QueryServiceStatusEx(  schService,  SC\_STATUS\_PROCESS\_INFO,  (LPBYTE)&ssp,  sizeof(SERVICE\_STATUS\_PROCESS),  &dwBytesNeeded ) )  {  printf("QueryServiceStatusEx failed (%d)\n", GetLastError());  goto stop\_cleanup;  }  if ( ssp.dwCurrentState == SERVICE\_STOPPED )  {  printf("Service stopped successfully.\n");  goto stop\_cleanup;  }  if ( GetTickCount() - dwStartTime > dwTimeout )  {  printf("Service stop timed out.\n");  goto stop\_cleanup;  }  }  // If the service is running, dependencies must be stopped first.  StopDependentServices();  // Send a stop code to the service.  if ( !ControlService(  schService,  SERVICE\_CONTROL\_STOP,  (LPSERVICE\_STATUS) &ssp ) )  {  printf( "ControlService failed (%d)\n", GetLastError() );  goto stop\_cleanup;  }  // Wait for the service to stop.  while ( ssp.dwCurrentState != SERVICE\_STOPPED )  {  Sleep( ssp.dwWaitHint );  if ( !QueryServiceStatusEx(  schService,  SC\_STATUS\_PROCESS\_INFO,  (LPBYTE)&ssp,  sizeof(SERVICE\_STATUS\_PROCESS),  &dwBytesNeeded ) )  {  printf( "QueryServiceStatusEx failed (%d)\n", GetLastError() );  goto stop\_cleanup;  }  if ( ssp.dwCurrentState == SERVICE\_STOPPED )  break;  if ( GetTickCount() - dwStartTime > dwTimeout )  {  printf( "Wait timed out\n" );  goto stop\_cleanup;  }  }  printf("Service stopped successfully\n");  stop\_cleanup:  CloseServiceHandle(schService);  CloseServiceHandle(schSCManager);  }  BOOL \_\_stdcall StopDependentServices()  {  DWORD i;  DWORD dwBytesNeeded;  DWORD dwCount;  LPENUM\_SERVICE\_STATUS lpDependencies = NULL;  ENUM\_SERVICE\_STATUS ess;  SC\_HANDLE hDepService;  SERVICE\_STATUS\_PROCESS ssp;  DWORD dwStartTime = GetTickCount();  DWORD dwTimeout = 30000; // 30-second time-out  // Pass a zero-length buffer to get the required buffer size.  if ( EnumDependentServices( schService, SERVICE\_ACTIVE,  lpDependencies, 0, &dwBytesNeeded, &dwCount ) )  {  // If the Enum call succeeds, then there are no dependent  // services, so do nothing.  return TRUE;  }  else  {  if ( GetLastError() != ERROR\_MORE\_DATA )  return FALSE; // Unexpected error  // Allocate a buffer for the dependencies.  lpDependencies = (LPENUM\_SERVICE\_STATUS) HeapAlloc(  GetProcessHeap(), HEAP\_ZERO\_MEMORY, dwBytesNeeded );    if ( !lpDependencies )  return FALSE;  \_\_try {  // Enumerate the dependencies.  if ( !EnumDependentServices( schService, SERVICE\_ACTIVE,  lpDependencies, dwBytesNeeded, &dwBytesNeeded,  &dwCount ) )  return FALSE;  for ( i = 0; i < dwCount; i++ )  {  ess = \*(lpDependencies + i);  // Open the service.  hDepService = OpenService( schSCManager,  ess.lpServiceName,  SERVICE\_STOP | SERVICE\_QUERY\_STATUS );  if ( !hDepService )  return FALSE;  \_\_try {  // Send a stop code.  if ( !ControlService( hDepService,  SERVICE\_CONTROL\_STOP,  (LPSERVICE\_STATUS) &ssp ) )  return FALSE;  // Wait for the service to stop.  while ( ssp.dwCurrentState != SERVICE\_STOPPED )  {  Sleep( ssp.dwWaitHint );  if ( !QueryServiceStatusEx(  hDepService,  SC\_STATUS\_PROCESS\_INFO,  (LPBYTE)&ssp,  sizeof(SERVICE\_STATUS\_PROCESS),  &dwBytesNeeded ) )  return FALSE;  if ( ssp.dwCurrentState == SERVICE\_STOPPED )  break;  if ( GetTickCount() - dwStartTime > dwTimeout )  return FALSE;  }  }  \_\_finally  {  // Always release the service handle.  CloseServiceHandle( hDepService );  }  }  }  \_\_finally  {  // Always free the enumeration buffer.  HeapFree( GetProcessHeap(), 0, lpDependencies );  }  }  return TRUE;  } |

## [修改服务的 DACL](https://learn.microsoft.com/zh-cn/windows/win32/services/modifying-the-dacl-for-a-service)

|  |
| --- |
| [服务控制程序](https://learn.microsoft.com/zh-cn/windows/win32/services/service-control-programs)可以创建或修改与服务关联的 DACL 来控制访问。 若要检索与服务对象关联的 DACL，请使用 [**QueryServiceObjectSecurity**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-queryserviceobjectsecurity) 函数。 若要设置 DACL，请使用 [**SetServiceObjectSecurity**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winsvc/nf-winsvc-setserviceobjectsecurity) 函数。 在从系统中删除服务之前，对与服务对象关联的 [**SECURITY\_DESCRIPTOR**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winnt/ns-winnt-security_descriptor) 所做的任何更改都是永久性的。  以下示例创建并设置服务的新 DACL。 该代码将一个访问控制项 (ACE) 合并到服务的现有 DACL。 新的 ACE 授予来宾帐户启动、停止、删除和READ\_CONTROL对指定服务的访问权限。 可以通过传递给 [**BuildExplicitAccessWithName**](https://learn.microsoft.com/zh-cn/windows/desktop/api/aclapi/nf-aclapi-buildexplicitaccesswithnamea) 函数的 AccessPermissions 参数修改对服务的访问权限。  szSvcName 变量是包含服务名称的全局变量。 有关设置此变量的完整示例，请参阅 [SvcControl.cpp](https://learn.microsoft.com/zh-cn/windows/win32/services/svccontrol-cpp)。  C++复制  //  // Purpose:  // Updates the service DACL to grant start, stop, delete, and read  // control access to the Guest account.  //  // Parameters:  // None  //  // Return value:  // None  //  VOID \_\_stdcall DoUpdateSvcDacl()  {  EXPLICIT\_ACCESS ea;  SECURITY\_DESCRIPTOR sd;  PSECURITY\_DESCRIPTOR psd = NULL;  PACL pacl = NULL;  PACL pNewAcl = NULL;  BOOL bDaclPresent = FALSE;  BOOL bDaclDefaulted = FALSE;  DWORD dwError = 0;  DWORD dwSize = 0;  DWORD dwBytesNeeded = 0;  // Get a handle to the SCM database.    schSCManager = OpenSCManager(  NULL, // local computer  NULL, // ServicesActive database  SC\_MANAGER\_ALL\_ACCESS); // full access rights    if (NULL == schSCManager)  {  printf("OpenSCManager failed (%d)\n", GetLastError());  return;  }  // Get a handle to the service  schService = OpenService(  schSCManager, // SCManager database  szSvcName, // name of service  READ\_CONTROL | WRITE\_DAC); // access    if (schService == NULL)  {  printf("OpenService failed (%d)\n", GetLastError());  CloseServiceHandle(schSCManager);  return;  }  // Get the current security descriptor.  if (!QueryServiceObjectSecurity(schService,  DACL\_SECURITY\_INFORMATION,  &psd, // using NULL does not work on all versions  0,  &dwBytesNeeded))  {  if (GetLastError() == ERROR\_INSUFFICIENT\_BUFFER)  {  dwSize = dwBytesNeeded;  psd = (PSECURITY\_DESCRIPTOR)HeapAlloc(GetProcessHeap(),  HEAP\_ZERO\_MEMORY, dwSize);  if (psd == NULL)  {  // Note: HeapAlloc does not support GetLastError.  printf("HeapAlloc failed\n");  goto dacl\_cleanup;  }    if (!QueryServiceObjectSecurity(schService,  DACL\_SECURITY\_INFORMATION, psd, dwSize, &dwBytesNeeded))  {  printf("QueryServiceObjectSecurity failed (%d)\n", GetLastError());  goto dacl\_cleanup;  }  }  else  {  printf("QueryServiceObjectSecurity failed (%d)\n", GetLastError());  goto dacl\_cleanup;  }  }  // Get the DACL.  if (!GetSecurityDescriptorDacl(psd, &bDaclPresent, &pacl,  &bDaclDefaulted))  {  printf("GetSecurityDescriptorDacl failed(%d)\n", GetLastError());  goto dacl\_cleanup;  }  // Build the ACE.  BuildExplicitAccessWithName(&ea, TEXT("GUEST"),  SERVICE\_START | SERVICE\_STOP | READ\_CONTROL | DELETE,  SET\_ACCESS, NO\_INHERITANCE);  dwError = SetEntriesInAcl(1, &ea, pacl, &pNewAcl);  if (dwError != ERROR\_SUCCESS)  {  printf("SetEntriesInAcl failed(%d)\n", dwError);  goto dacl\_cleanup;  }  // Initialize a new security descriptor.  if (!InitializeSecurityDescriptor(&sd,  SECURITY\_DESCRIPTOR\_REVISION))  {  printf("InitializeSecurityDescriptor failed(%d)\n", GetLastError());  goto dacl\_cleanup;  }  // Set the new DACL in the security descriptor.  if (!SetSecurityDescriptorDacl(&sd, TRUE, pNewAcl, FALSE))  {  printf("SetSecurityDescriptorDacl failed(%d)\n", GetLastError());  goto dacl\_cleanup;  }  // Set the new DACL for the service object.  if (!SetServiceObjectSecurity(schService,  DACL\_SECURITY\_INFORMATION, &sd))  {  printf("SetServiceObjectSecurity failed(%d)\n", GetLastError());  goto dacl\_cleanup;  }  else printf("Service DACL updated successfully\n");  dacl\_cleanup:  CloseServiceHandle(schSCManager);  CloseServiceHandle(schService);  if(NULL != pNewAcl)  LocalFree((HLOCAL)pNewAcl);  if(NULL != psd)  HeapFree(GetProcessHeap(), 0, (LPVOID)psd);  } |