**Windows服务**

创建在 Windows 中的可长时间运行的可执行应用程序。

这些服务可以在计算机启动时自动启动，可以暂停和重新启动而且不显示任何用户界面。它非常适合在[服务器](https://cloud.tencent.com/act/pro/promotion-cvm?from_column=20065&from=20065)上使用，或为了不影响在同一台电脑上工作的其他用户需要长时间运行功能时使用，或者是随开机就启动后台默默干活的应用。

通过运行界面，输入services.msc可打开windows自带的服务管理界面对服务进行控制。

一个服务不管有没有被运行，都在你的硬盘里,只有当它真正被运行时，操作系统就会真正给它分配内存、CPU时间片等资源，这一次运行就对应一个“进程”。

**服务管理**

可以使用SC命令对服务进行管理，SC 是用于与服务控制管理器和服务进行通信的命令行程序。

如下示例：

代码语言：CMD

#创建服务

sc create myservice binpath= "c:\mywin32.exe" displayname= "myService"

sc delete myservice

sc start myservice

sc stop myservice

随便写一个hello.exe以这种方式加进去行不行？

结论是可以添加进去，在服务管理界面也能看到。但是没什么意义，因为无法启动也无法控制，它也不会自动运行。

服务需要符合一定的写法。

**基本写法**

需要的有：

服务入口函数：

void WINAPI ServiceMain(int argc, char\* argv[])

 服务控制函数（暂停、关机一些命令的控制）

ServiceCtrlHandler(DWORD dwControl, DWORD dwEventType, LPVOID lpEventData, LPVOID lpContext)

服务状态结构体

SERVICE\_STATUS m\_ServiceStatus;

注册服务控制函数

RegisterServiceCtrlHandlerEx("myService",lpHandlerProc,myContext)

设置服务状态函数

SetServiceStatus(m\_ServiceStatusHandle, &m\_ServiceStatus)

服务进入表，设置服务程序的入口

SERVICE\_TABLE\_ENTRY DispatchTable[2]

服务控制分发

 StartServiceCtrlDispatcher(DispatchTable)

基本上完成以上这些，就可以创建一个简单的windows服务了，熟悉下套路还是很简单的。编译完成后生成的.exe可执行文件不能直接运行，使用上面的sc命令来创建和启动服务。

还有一些稍复杂的点用法，可以完成应用自身对服务的创建，启动和停止等管理。

比如windows上的redis服务，它通过：

./redis-server.exe  --service-install redis.windows-service.conf 就可以把自己以服务的方式运行和启动。是它的源码内部实现了服务的创建，暂停，启动等方法。如：ServiceInstall，ServiceStart，ServiceStop等。参见redis源码中的Win32\_service.cpp。

源码地址：[https://github.com/tporadowski/redis](https://cloud.tencent.com/developer/tools/blog-entry?target=https%3A%2F%2Fgithub.com%2Ftporadowski%2Fredis&source=article&objectId=1994430)

**简单示例**

代码语言：c

#include <Windows.h>

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <helper.h>

#include <logging.h>

#include <redisclient.h>

using namespace std;

const int SLEEP\_TIME = 5000; // 延迟时间

BOOL bFlag = TRUE; // 标记循环是否结束

SERVICE\_STATUS\_HANDLE m\_ServiceStatusHandle; // 服务状态句柄

SERVICE\_STATUS m\_ServiceStatus; // 服务状态结构体，保存服务程序的一些信息

// 命令行参数可加可不加

void WINAPI ServiceMain(int argc, char\* argv[]);// 服务程序入口函数

//void WINAPI ServiceCtrlHandler(DWORD Opcode); // 服务控制函数（暂停、关机一些命令的控制）

DWORD WINAPI ServiceCtrlHandler(DWORD dwControl, DWORD dwEventType, LPVOID lpEventData, LPVOID lpContext); // 服务控制函数

LPVOID myContext;

void WINAPI ServiceMain(int argc, char\* argv[])

{

MEMORYSTATUS memstatus; // 内存状态结构体，存储内存的一些信息

char str[100]; // 存储一个字符串

size\_t availmb; // 将获取到的内存大小 B --> MB转换

// 初始化服务状态

m\_ServiceStatus.dwServiceType = SERVICE\_WIN32; // Win32类型

m\_ServiceStatus.dwCurrentState = SERVICE\_START\_PENDING; // 当前状态为开始等待

m\_ServiceStatus.dwControlsAccepted = SERVICE\_ACCEPT\_PAUSE\_CONTINUE|SERVICE\_ACCEPT\_STOP; // 接受关机与暂停两种控制

m\_ServiceStatus.dwWin32ExitCode = 0; // 下面四个默认都为0

m\_ServiceStatus.dwServiceSpecificExitCode = 0;

m\_ServiceStatus.dwCheckPoint = 0;

m\_ServiceStatus.dwWaitHint = 0;

//重要

m\_ServiceStatus.dwControlsAccepted |= SERVICE\_ACCEPT\_PRESHUTDOWN;

//注册服务控制

//m\_ServiceStatusHandle = RegisterServiceCtrlHandler("myService", ServiceCtrlHandler);

m\_ServiceStatusHandle = RegisterServiceCtrlHandlerEx("myService",ServiceCtrlHandler,myContext);

if (m\_ServiceStatusHandle == 0) // 判断是否成功执行

{

LOGGING\_DEBUG("RegisterServiceCtrlHandle failed"); // 错误信息写入文件

return;

}

LOGGING\_DEBUG("RegisterServiceCtrlHandle success"); // 成功信息写入文件

m\_ServiceStatus.dwCurrentState = SERVICE\_RUNNING; // 成功运行则把状态设置为运行状态

SetServiceStatus(m\_ServiceStatusHandle, &m\_ServiceStatus); // 设置状态

bFlag = TRUE; // 循环条件为真

memset(str, 0, 100); // 初始化字符串中的数据

while (bFlag)

{

//todo,fixeme,服务空闲状态干什么事,这里假做记录内存信息

GlobalMemoryStatus(&memstatus); // 获取内存状态信息

availmb = memstatus.dwAvailPhys / 1024 / 1024; // 将获取的字节转化为M

sprintf\_s(str, 100, "availmb memory is %d MB ", availmb); // 格式化字符串

LOGGING\_DEBUG(str); // 写入文件中

Sleep(SLEEP\_TIME); // 延迟五秒

}

LOGGING\_DEBUG("service stopped"); // 结束循环后，发送一消息给文件

}

void doYourWork(){

//ostringstream out;

//out << "RedisClient isConnect = " << ret;

//LOGGING\_DEBUG((char\*)out.str().c\_str());

}

int WriteToLog(char\* str) //自定义的写日志函数

{

FILE\* pfile;

fopen\_s(&pfile,"C:/mylog.log","a+"); //已追加的方式打开文件；fopen\_s函数的安全性更高；

if (pfile==NULL)

{

return -1;

}

fprintf\_s(pfile,"%s\n",str); //向文件中写入字符串

fclose(pfile); //关闭打开的文件

return 0;

}

//注册服务控制处理

DWORD ServiceCtrlHandler(DWORD dwControl, DWORD dwEventType, LPVOID lpEventData, LPVOID lpContext){

switch( dwControl ){

//如果是关机事件,则...

case SERVICE\_CONTROL\_PRESHUTDOWN:

LOGGING\_DEBUG("myService. SERVICE\_CONTROL\_PRESHUTDOWN");

doYourWork();

//ShutdownBlockReasonCreate(hwnd, (("waiting for user response")));

break;

case SERVICE\_CONTROL\_STOP: // 暂停控制

bFlag = FALSE; // 循环标志为FASE

m\_ServiceStatus.dwCurrentState = SERVICE\_STOPPED;// 状态设置为停止

LOGGING\_DEBUG("myService. SERVICE\_CONTROL\_STOP");

break;

case SERVICE\_CONTROL\_CONTINUE: // 继续

LOGGING\_DEBUG("myService. SERVICE\_CONTROL\_CONTINUE");

break;

case SERVICE\_CONTROL\_SHUTDOWN:// 关机控制

bFlag = FALSE; // 循环标志为FASE

m\_ServiceStatus.dwCurrentState = SERVICE\_STOPPED;// 状态设置为停止

LOGGING\_DEBUG("myService. SERVICE\_CONTROL\_SHUTDOWN");

break;

default:

ostringstream oStr;

oStr << "lpHandlerProc dwControl=" << dwControl;

LOGGING\_DEBUG((char\*)oStr.str().c\_str());

break;

}

// 设置服务状态

SetServiceStatus(m\_ServiceStatusHandle, &m\_ServiceStatus);

return 0;

}

int main(){

cout<<"hello myService start"<<endl;

LOGGING\_DEBUG(" myService start...");

//为当前调用的进程设置关闭的参数，此函数为进程设置一个相对于系统中其它进程的关闭顺序

int ret = SetProcessShutdownParameters(0x3FF,0);

if(ret == 0){

LOGGING\_ERROR("SetProcessShutdownParameters error");

}

// 定义服务进入表，设置服务程序的入口函数

SERVICE\_TABLE\_ENTRY DispatchTable[2];

DispatchTable[0].lpServiceName = "myService"; // 服务程序名称

DispatchTable[0].lpServiceProc = (LPSERVICE\_MAIN\_FUNCTION)ServiceMain; // 服务程序入口

DispatchTable[1].lpServiceName = NULL;

DispatchTable[1].lpServiceProc = NULL;

// 开始服务控制分布

StartServiceCtrlDispatcher(DispatchTable);

return 0;

}

**引用**

[Windows 服务程序(一) - 走看看](https://cloud.tencent.com/developer/tools/blog-entry?target=http%3A%2F%2Ft.zoukankan.com%2FM-Anonymous-p-9380772.html&source=article&objectId=1994430)

[正在关闭 - Win32 apps | Microsoft Docs](https://cloud.tencent.com/developer/tools/blog-entry?target=https%3A%2F%2Fdocs.microsoft.com%2Fzh-cn%2Fwindows%2Fwin32%2Fshutdown%2Fshutting-down&source=article&objectId=1994430)

[关于服务 - Win32 apps | Microsoft Docs](https://cloud.tencent.com/developer/tools/blog-entry?target=https%3A%2F%2Fdocs.microsoft.com%2Fzh-cn%2Fwindows%2Fwin32%2Fservices%2Fabout-services&source=article&objectId=1994430)