实 验 十三　Windows 服务程序

13.1　实验目的

理解 Windows 服务程序工作机制，掌握使用 VS12 工具开发和调试服务程序的方法。

13.2　服务程序介绍

显示界面是用户与程序交互的载体，程序通过显示界面获取输入并进行结果输出。界面交互在直接面向用户的程序中很重要，还有一些 Windows 程序运行时特点如下：

1. 自动完成事先安排的系统任务。
2. 无需人工操作或不适合人工操作。
3. 往往运行在机房或远端。
4. 提供网络或数据的服务端。

这些程序随机器启动而自动运行，由服务控制中心管理这就是服务程序。服务程序在客户机/服务器软件协作模式中扮演服务器角色，例如用户使用浏览器下载网页的 Browser 软件是客户端，而提供网页的服务器程序是服务端。服务程序按照逻辑执行机器监视和管理任务，无须用户交互操作，UNIX/Linux 系统中与服务程序扮演类似角色的进程称为守护进程。

Windows 服务运行包含三种组件：

1. 服务应用程序。
2. 服务控制程序 (SCP)。
3. 服务控制管理 (SCM)。

服务应用程序完成特定服务任务由 SCP 和 SCM 进行管理，通过控制面板 -> 管理工具 -> 服务打开服务控制管理 (SCM)，SCM 管理机器中所有服务程序。只有经过注册的服务程序接受 SCM 管理才能执行，SCM 收到程序的注册消息后将会在注册表中添加一项，注册表项包括服务程序的路径和配置参数。服务控制程序 (SCP) 用来启动、停止、配置服务程序，windows 操作系统提供了内置的 SCP 程序，某些服务应用程序也可以有自己的 SCP 来配置服务的运行参数。服务应用程序具有特定的代码能够从 SCM 接收命令，也能将服务程序当前状态返回到 SCM。

服务程序的管理和运行由内核函数实现，与服务相关函数实现在 Advapi32.dll 模块中，表13-1列出了 Windows 内核服务相关函数，使用 C++ 语言开发服务程序的读者直接使用此表中的函数。

服务程序的启动与关闭都是由系统自动调用的，在机器启动时由 Winlogon 进程启动 SCM，SCM 的程序文件是 \Windows\System32\Services.exe。当进程 WinLogon 调用了

表 13-1 服务程序 API

函数名	函数功能
CreateService	注册一个服务程序。
ControlService	请求 SCM 向服务程序发送控制命令。
DeleteService	标记一个服务程序为删除，SCM 将会从注册表中删服务项。
OpenSCManager	服务控制程序首先要使用 OpenSCManager 函数建立与 SCM 的通信通道。
OpenService	打开一个已知服务程序，取得其句柄。
QueryServiceStatus	获取指定服务程序状态信息。
StartService	启动服务一个服务程序。
SetServiceStatus	更新服务程序状态信息。

ExitWindowsEx 函数时，ExitWindowsEx 向 Csrss 进程发送消息。Csrss 进程调用 shutdown 过程，它将遍历所有的进程并通知它们系统正在关闭，Csrss 对每个进程会等待结束一定时间，SCM 关闭处理函数会向每个服务程序发送关闭消息。SCM 会等待每个服务程序的结束，Csrss 等待 SCM 的结束，如果服务程序不能在一定时间内关闭，而系统关机过程会继续，那么服务程序就无法完美完成关闭任务了。

13.2.1 .NET 服务类

使用 Windows 的服务相关 API 来创建服务程序的难度很大，.Net 框架提供服务程序类大大地简化服务程序的创建和控制，和 Windows 服务程序相关的命名空间涉及到以下两个：System.ServiceProcess 和 System.Diagnostics。创建一个基本的 Windows 服务程序，需要使用.Net 框架 System.ServiceProcess 命名空间中的四个类：ServiceBase、ServiceInstaller、ServiceProcessInstaller 以及 ServiceController，它们的关系结构可见图 13-1。

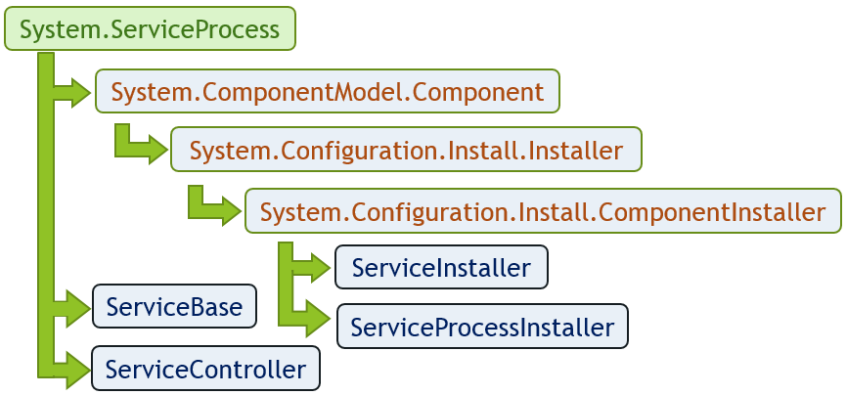

图 13-1 System.ServiceProcess 命名空间中的四个类

13.3 创建快捷方式的服务程序

本小节示例项目使用 VS12 工具基于.NET 平台类编写服务程序，它按时间间隔循环方式创建 mspaint.exe 程序的快捷方式。

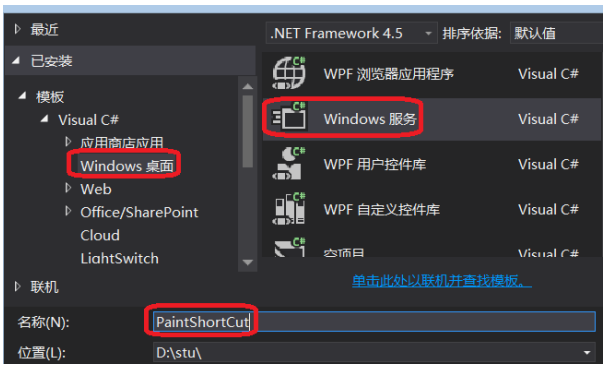

图 13-2 新建项目 —windows 服务

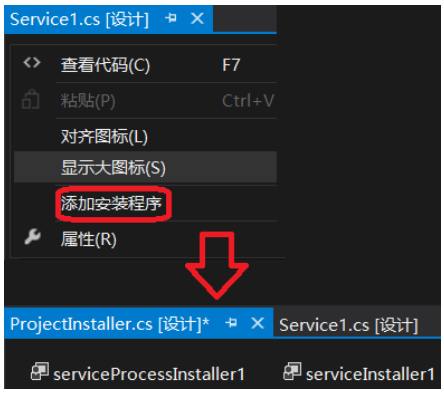

图 13-3 服务添加安装程序

13.3.1 程序准备

Windows 服务程序与控制台程序，窗体应用程序都是可执行代码，但是运行方式不同。服务程序典型情况没有用户交互界面，也较少具有输入输出功能，有些控制台或窗体的输入输出函数无法使用，服务程序编写方法和运行调试与普通程序都不同。使用 VS12 工具新建项目，项目类型是 Visual C# 下的 Windows 桌面，模板类型选择 Windows 服务，项目名称为 PaintShortCut，位置是 d:\stu 目录；参考图为 13-2，点击确定即创建一个新服务应用程序。

VS12 根据模板建立一个 Service1 的类文件，在 Service1 的设计状态下使用右键菜单，找到"添加安装程序"菜单项，项目将添加一个名为 ProjectInstaller 的类文件，在 ProjectInstaller 的设计视图中可以看到两个对象，其中 serviceInstaller1 对象是 System.ServiceProcess.ServiceInstaller 类型，serviceProcessInstaller1 对象是 System.ServiceProcess.ServiceProcessInstaller 类型，可参看图图 13-3 。新建的服务程序现在包含三个主要文件，Program.cs 文件负责项目的启动，Service1.cs 定义的类继承自 ServiceBase 类；ProjectInstaller1.cs 定义了 ServiceInstaller 类和 ServiceProcessInstaller 类型对象，类关系可参考图 13-1所示。

表13-2列出在设计时 ServiceInstaller 类和 ServiceProcessInstaller 类型对象属性：

表 13-2 服务程序的常规属性设置

SCM 中名称	控件属性	属性值
服务名称	serviceInstaller1.ServiceName	PaintShortCut
显示名称	serviceInstaller1.DisplayName	PaintShortCutBuild
描述	serviceInstaller1.Description	在指定目录创建 Paint 的多个快捷方式。
启动类型	serviceInstaller1.StartType	Automatic
登录身份	serviceProcessInstaller1.Account	LocalSystem

图13-4显示系统已经注册的服务程序在 SCM 中和常规属性，serviceInstaller1 的 StartType 设置服务的启动方式，本示例中选为 Automatic(自动启动)；Description 是对服务的文本描述信息，ServiceName 指示服务的名称。设置服务程序项目的输出路径为 d:\stu\，使用 F6 命令对其编译生成名称为 PaintShortCut.exe 的服务程序。

服务程序的启动非常特殊，首先须由安装实用工具（如 InstallUtil.exe）将其安装到机器中，

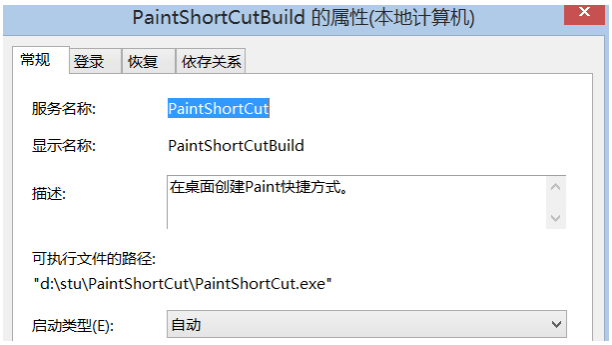

图 13-4 服务程序的常规和登录属性

再在服务管理界面启动服务。InstallUtil.exe 文件所在目录为 C:\Windows\Microsoft.NET\Framework\v4.0.303 InstallUtil.exe 工具安装本服务的方法如下：

installutil D:\stu\PaintShortCut\PaintShortCut.exe

卸载服务程序的方法是在上述命令中添加/u 参数，形如：

installutil /u D:\stu\PaintShortCut\PaintShortCut.exe。

13.3.2 服务程序代码实现

生成的服务程序虽然在结构上是完整的，但是一空壳子没有任何实际代码，只有添加任务代码才能发挥程序功能。服务代码编写在 ServiceBase 派生的中类，放置在重载的 OnStart 和 OnStop 方法中，本服务程序将创建快捷方式功能安排在工作线程中。下面是 Service1 类中变量定义：

```
public static ManualResetEvent terminateThrE;
public static string paintPath;
public static int shortCutIndex = 1;
```

线程使用了 Windows Script Host Object Model 的 COM 组件，通过创建 WshShell 对象来生成 mspaint.exe 的快捷方式参考代码如下：

```
static void CreatePaintShortCut()
{
  paintPath = @"C:\Windows\system32\mspaint.exe";
  WshShell shell = new WshShell();
  while (!terminateThrE.WaitOne(2000))
  {
    if(shortCutIndex>30)
    {
    break;
    }
    IWshShortcut shortcut = (IWshShortcut)shell.CreateShortcut(
    string.Format("D:\\ 画图程序 {0}.lnk",shortCutIndex));
    shortcut.TargetPath = @"C:\Windows\system32\mspaint.exe";
```

```
        shortcut.WorkingDirectory = System.Environment.CurrentDirectory;
        shortcut.WindowStyle = 1;
        shortcut.Description = " 启动画图程序";
        shortcut.IconLocation = @"C:\Windows\system32\mspaint.exe,0";
        shortcut.Save();
        shortCutIndex++;
      }
    }
```

在重载的 OnStart 事件中启动线程：

```
  protected override void OnStart(string[] args)
  {
    terminateThrE = new ManualResetEvent(false);
    Thread checkDesk = new Thread(new ThreadStart(CreatePaintShortCut));
    checkDesk.IsBackground = true;
    checkDesk.Start();
  }
```

在重载的 OnStop 事件中设置低层事件变量终止状态，使线程自行结束：

```
  protected override void OnStop()
  {
    terminateThrE.Set();
  }
```

13.3.3 服务程序的调试与运行

服务程序的启动过程程序较繁琐，包括下面几个步骤：

1. 使用 F6 生成服务程序 PaintShortCut.exe；
2. 使用 InstallUtil 工具安装服务程序；
3. 打开或者刷新 SCM，找到安装的 PaintShortCut 服务；
4. 启动服务程序；
5. 如果发生异常或程序逻辑错误则需要停止服务程序；
6. 重新编译代码并重复上述步骤，直至服务运行正常；

服务程序运行过程的错误信息可使用 EventLog.WriteEntry 方法输出信息到系统的事件查看器中查看。

13.4 关闭机器服务程序

本小节编写一个关闭机器的服务程序，新建 Windows 服务项目并命名为 MachineDaemon，位置为 D:\stu。服务程序运行逻辑是检测机器的运行时间，如果当前运行时间是晚上 23 点至早上 6 点之间，则关闭机器。在 Service1.cs 文件中添加下面的代码：

```
      [StructLayout(LayoutKind.Sequential, Pack = 1)]
  internal struct TokPriv1Luid
```

```
{
    public int Count;
    public long Luid;
    public int Attr;
}
//声明要使用的 API
[DllImport("kernel32.dll", ExactSpelling = true)]
internal static extern IntPtr GetCurrentProcess();
[DllImport("advapi32.dll", ExactSpelling = true, SetLastError = true)]
internal static extern bool OpenProcessToken(IntPtr h, int acc, ref IntPtr phtok);
[DllImport("advapi32.dll", SetLastError = true)]
internal static extern bool LookupPrivilegeValue(string host, string name, ref long pluid);
[DllImport("advapi32.dll", ExactSpelling = true, SetLastError = true)]
internal static extern bool AdjustTokenPrivileges(IntPtr htok, bool disall,
ref TokPriv1Luid newst, int len, IntPtr prev, IntPtr relen);
[DllImport("user32.dll", ExactSpelling = true, SetLastError = true)]
//定义要使用的常量
internal static extern bool ExitWindowsEx(int DoFlag, int rea);
internal const int SE_PRIVILEGE_ENABLED = 0x00000002;
internal const int TOKEN_QUERY = 0x00000008;
internal const int TOKEN_ADJUST_PRIVILEGES = 0x00000020;
internal const string SE_SHUTDOWN_NAME = "SeShutdownPrivilege";
internal const int EWX_LOGOFF = 0x00000000;
internal const int EWX_SHUTDOWN = 0x00000001;
internal const int EWX_REBOOT = 0x00000002;
internal const int EWX_FORCE = 0x00000004;
internal const int EWX_POWEROFF = 0x00000008;
internal const int EWX_FORCEIFHUNG = 0x00000010;
//关闭机器函数
private static void DoExitWin(int DoFlag)
{
    bool ok;
    TokPriv1Luid tp;
    IntPtr hproc = GetCurrentProcess();
    IntPtr htok = IntPtr.Zero;
    ok = OpenProcessToken(hproc, TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY,
ref htok);
    tp.Count = 1;
    tp.Luid = 0;
    tp.Attr = SE_PRIVILEGE_ENABLED;
```

```
    ok = LookupPrivilegeValue(null, SE_SHUTDOWN_NAME, ref tp.Luid);
    ok = AdjustTokenPrivileges(htok, false, ref tp, 0, IntPtr.Zero, IntPtr.Zero);
    ok = ExitWindowsEx(DoFlag, 0);
  }
  public static int bootMinute = 0;
  public static ManualResetEvent e_1;
  public static ManualResetEvent e_2;
```

关闭机器使用 API 是 ExitWindowsEx 函数，这个函数根据传入常量值参数用来关闭系统，或者重启系统，这个函数能够向所有应用程序发送 WM_QUERYENDSESSION 消息来终止进程；这个函数的执行需要获取机器的 SE_SHUTDOWN_NAME 权限，程序没有足够权限时执行 ExitWindowsEx 函数不会成功；通过 LookupPrivilegeValue 函数和 AdjustTokenPrivileges 函数，进程可获取足够权限。服务程序的代码是在 OnStart 函数要执行的，下面是完整的代码。

```
    protected override void OnStart(string[] args)
  {
    //向事件查看器写入日志
    EventLog.WriteEntry("MachineDaemon OnStart", DateTime.Now.ToLongTimeString());
    bootMinute = 3;
    e_1 = new ManualResetEvent(false);
    e_2 = new ManualResetEvent(false);
    e_1.Reset();
    e_2.Reset();
    do
    {//首先等待机器三分钟
      if ((Environment.TickCount / 180000) > bootMinute)
      {
        e_1.Set();
      }
      //等待 20 秒的时间
      e_1.WaitOne(20000,false);
      //检测等待事件的状态，没有被设置则继续循环
    } while (e_1.WaitOne(1,false));
    EventLog.WriteEntry("MachineDaemon check three minutes", DateTime.Now.ToLongTimeString());
    DateTime cur_time;
    do
    { //每分钟检查一次
      cur_time = DateTime.Now;
      if ((cur_time.Hour < 7) || (cur_time.Hour > 21))
      {//当前时间介于 22 点至 6 点，则关机
        e_2.Set();
      }
```

```
        //等待 30 秒的时间
        e_2.WaitOne(30000,false);
        //检测等待事件的状态，没有被设置则继续循环
    } while (!e_2.WaitOne(1,false));
    EventLog.WriteEntry("MachineDaemon DoExitWin", DateTime.Now.ToLongTimeString());
    //执行关闭机器函数
    DoExitWin(EWX_FORCE | EWX_POWEROFF);
}
```

在这个服务程序中，有两个部分，第一阶段检查机器的运行时间是否已经超过三分钟；第二阶段是检测机器的当前时间，判断是否处于 23 点到 6 点这个时间段，满足条件时即调用 DoExitWin 函数来关闭计算机。在 Windows 环境中编写与时间循环等待相关的功能代码时，必须避免程序忙等检测的不良方式，比较有效率的方式是使用 ManualResetEvent 对象的 WaitOne 方法，这种方式不会使计算机进入忙等，而浪费 CPU 时间。完整代码的服务程序使用编译命令，生成 MachineDaemon.exe，进入命令行状态，切换到 installutil 的目录，执行命令 installutil D:\min\MachineDaemon\MachineDaemon.exe 将这个程序安装为系统服务，在管理工具的服务窗体可见到刚刚安装的这个服务如图 13-5。这个服务程序会在机器启动三分钟后，检测当前时间段是否为设定好的时间段，满足设定时间段会关闭计算机，服务程序开发完成。要注意哦如果机器时钟刚好处于设定的时间段，则会在开机三分钟后马上关机，读者可将这个时间设得稍长一些。

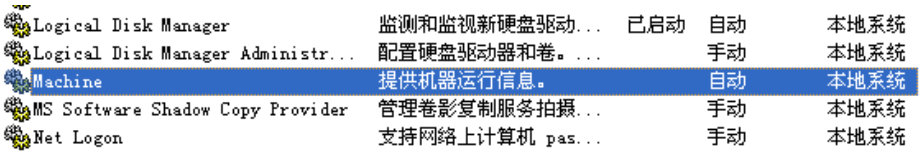

图 13-5 windows 服务列表 –Machine 服务

使用.NET FrameWorK 类开发服务程序比使用平台 API 方便，服务程序不具有用户交互能力，运行和调试有些繁琐，服务程序中与窗体界面相关的 API 都无法使用，对开发技巧有更高的要求。

13.5 作业

1. 完成自动生成快捷方式的服务程序。
2. 编写一个服务程序实现定时关机，关机任务可通过执行 ShutDown 命令实现。