外部程序管理器系统结构的设计与实现

宁　伟　（泰安师范专科学校数学与计算机科学系　271000）

摘要：EPM(External Program Manager)通过封装复杂的程序逻辑，实现了一个界面简洁的构件化系统，它可以方便地集成到应用程序中，从而简化了应用程序调用其他程序的复杂性。本文主要介绍 EPM 的系统结构与实现方法。

关键词：同步执行　进程句柄　进程标识　窗口句柄

在应用程序的开发过程中，有时需要利用其他的应用程序来完成某些特定功能或辅助功能，如字处理程序、计算器等。一种情况是，人们希望两个程序能同步执行，即在外部程序运行过程中，调用程序处于等待状态，当外部程序退出后，调用程序才得到控制权继续执行。另一种情况是，要求两个程序异步执行，从而能实现一定的交互。在这两种需求中，有三个问题需要解决：

①如何执行一个同步调用，即满足第一种情况；

②如何检测一个外部程序的状态，以防止应用程序多次加载；

③如何使应用程序重新获得焦点，并且恢复正常的窗口状态。

本文首先介绍外部程序管理器(EPM: External Program Manager)的系统结构与信息模型，然后探讨系统实现及上述问题的解决办法。

1. 系统框架

为实现对外部应用程序的灵活调用，需要调用大量的 Windows API 函数，EPM 通过把具体方法封装进一个构件，达到了简化编程与方便集成的目的。系统框架如图 1 所示。下面分别介绍各个部件：

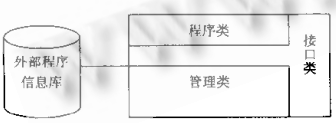

图 1　EPM 系统框架

(1) 外部应用程序信息库。采用关系数据库，存放注册应用程序的相关信息，是永久的外部程序对象，其信息模型如表 1 所示：

表 1　外部程序的信息模型

属 性	说 明
外部程序编号	外部程序的唯一性标识
程序注册名称	便于用户理解而定义的程序名称
程序命令行	程序的路径、名称与参数组成的字符串
执行方式	同步执行或者异步执行
列表标识	设置外部程序是否输出给调用程序，以便调用程序建立菜单项或填充列表框
注释	附加的说明信息或解释

(2) 程序类。程序类提供一个对象填充结构，封装外部程序的属性及其方法，每个外部程序都是程序类的一个实例。应用程序的启动与其状态检测等功能都封装在程序类中。

(3) 管理类。实现外部程序的注册与其信息的修改、删除等维护工作，以及确定可列表输出的程序项。

(4) 接口类。程序类与管理类是私有的，而接口类是外部可创建的。接口类提供相应的方法，输出管理类与程序类的功能。接口类首先实例化可列表的程序类，建立一个外部程序对象集合，调用程序通过 GetAvailablePrograms 方法获取程序列表；在用户指示运行某程序时，以程序编号调用 Execute 方法启动外部程序。

接口类的实现简化了编程接口，给使用 EPM 的开发人员以简洁的界面。

2. 系统实现

EPM 是在 Windows 环境下，以 Visual Basic 6.0 为开发工具实现的一个 ActiveX DLLs 构件。下面顺序介绍实现中的几个问题的解决方法：

(1)启动外部程序。在 VB 中，有两种调用外部应用程序的方法，一是利用 Shell 函数，它返回程序的进程标识，然后可以调用 API 函数 OpenProcess( )得到进程句

柄,因为在控制程序的同步执行或检测程序的状态中要用到进程句柄;一是直接调用 API 函数 CreateProcess( ),它将返回进程句柄与进程标识,只是其中涉及两个数据结构 STARTUPINFO 与 PROCESS - INFORMATION 较复杂。

①利用 Shell 函数与 OpenProcess 函数

```
'定义访问类型参数 dwAccess
'本常量可使获得的进程句柄用在 Wait 函数中等待进程终止
Private Const SYNCHRONIZE = &100000
'本常量可使获得的进程句柄用在 GetExitCodeProcess 函数和'GetPriorityClass 函数中去读进程对象中的信息
Private Const PROCESS _ QUERY _ INFORMATION = &H400
Private Declare Function OpenProcess Lib "kernel32.dll" _
(ByVal dwAccess As Long, ByVal fInherit As Integer, _
ByVal hObject As Long) As Long
ProcessID& = Shell(mProgram.CommandLine, _
    vbNormalFocus)
'调用 OpenProcess 函数以取得进程句柄,该句柄它将在函数 'WaitForSingleObject 和函数 GetExitCodeProcess 中使用
ProcessHandle& = OpenProcess
(PROCESS _ QUERY _ INFORMATION + SYNCHRONIZE,
  False, ProcessID&)
```

②利用 CreateProcess 函数

```
'STARTUPINFO 信息结构指定主窗口特性
Type STARTUPINFO
cb As Long '指定本结构的长度
lpReserved As String
lpDesktop As String
lpTitle As String
dwX As Long '确定窗口位置
dwY As Long
dwXSize As Long '确定窗口宽度和高度
dwYSize As Long
dwXCountChars As Long
dwYCountChars As Long
dwFillAttribute As Long
dwFlags As Long
wShowWindow As Integer '窗口显示方式
cbReserved2 As Long
lpReserved2 As Byte
hStdInput As Long '指定进程的标准输入句柄
hStdOutput As Long '指定进程的标准输出句柄
hStdError As Long '指定进程的标准错误句柄
End Type
'存放返回的进程句柄与标识及线程句柄与标识
Type PROCESS _ INFORMATION
hProcess As Long
hThread As Long
dwProcessId As Long
dwThreadId As Long
End Type
'定义一个进程优先级
Private Const NORMAL _ PRIORITY _ CLASS = &H20
'函数的定义
Private Declare Function CreateProcess Lib "kernel32" Alias _
" CreateProcessA " ( ByVal lpApplicationName As String,
ByVal lpCommandLine As String, lpProcessAttributes As
Any, lpThreadAttributes As Any, ByVal bInheritHandles As
Long, ByVal dwCreationFlags As Long, lpEnvironment As
Any, ByVal lpCurrentDriectory As String,
lpStartupInfo As STARTUPINFO,
lpProcessInformation As PROCESS _ INFORMATION)
As Long
'启动外部程序的具体实现
Dim pInfo As PROCESS _ INFORMATION
Dim sInfo As STARTUPINFO
Dim sNull As String
Dim lRes As Long
sInfo.cb = Len(sInfo)
lRes = CreateProcess(sNull, CmdStr $ , ByVal 0&,
```

```
ByVal 0&, _
        1&, NORMAL _ PRIORITY _ CLASS, ByVal
0&, sNull,
        sInfo, pInfo)
```

(2)控制程序的同步执行。为了控制进程的同步执行,调用 WaitForSingleObject 函数控制进程对象的状态,直到外部程序结束执行,才恢复父进程的控制权。

在进程对象终止后,父进程必须调用 CloseHandle 清除进程对象。

```
'定义一个等待时间
Private Const INFINITE = &HFFFF
Private Const SYN = True '同步执行
Private Declare Function WaitForSingleObject Lib "
kernel32"
    (ByVal hHandle As Long,
     ByVal dwMilliseconds As Long) As Long
Private Declare Function CloseHandle Lib "kernel32"
    (ByVal hObject As Long) As Long
'如果同步发起一个程序...
If ExecuteMode = SYN Then
' 等待外部程序结束
lRes = WaitForSingleObject(ProcessHandle, INFI-
NITE)
lRes = CloseHandle(ProcessHandle)
End If
```

(3)检测程序的状态。如果发起异步执行的外部程序,当用户再次使用它时,我们必须检查其状态,以防止它的多次加载。使用 GetExitCodeProcess 可检测程序是否仍在运行,如果返回 STILL _ ACTIVE,则可调用 VB 的 AppActivate 语句使其获得焦点。但是,AppActivate 不能使窗口程序从最小化状态恢复到正常状态,可以使用本节(4)介绍的 GetWindowHandle 函数取得窗口句柄,然后再调用 ShowWindow 函数。

```
'本常量定义窗口显示状态为正常
Private Const SW _ SHOWNORMAL = 1
Private Declare Function GetExitCodeProcess Lib "ker-
nel32"
        (ByVal hProcess As Long, _
        lpExitCode As Long) As Long
Private Declare Function ShowWindow Lib "user32"
(ByVal hwnd As Long, ByVal nCmdShow As Long)
As Long
'确定程序状态
lRes = GetExitCodeProcess(ProcessHandle, mCode)
If mCode = STILL _ ACTIVE Then
AppActivate ProcssID, True
lRes = ShowWindow(GetWindowHandle(Proces-
sID),
        SW _ SHOWNORMAL)
End If
```

(4)取得程序的窗口句柄。通过枚举窗口的方法,确定进程的主窗口。

```
'定义用到的常量与函数
Private Const GW _ HWNDFIRST = 0
Private Const GW _ HWNDNEXT = 2
Private Declare Function GetParent Lib "user32"
    (ByVal hwnd As Long) As Long
Private Declare Function GetWindow Lib "user32"
(ByVal hwnd As Long, ByVal wCmd As Long) As
Long
Private Declare Function GetWindowThreadProcessId
    Lib "user32" (ByVal hwnd As Long,
      lpdwProcessId As Long) As Long
'函数 GetWindowHandle 是取得进程的窗口句柄的
具体实现
Function GetWindowHandle (ByVal ProcessID As
Long,
        ByVal Apphwnd As Long) As Long
Dim mhwnd As Long
Dim mProcID As Long
Dim lRes as long

mhwnd = GetWindow(Apphwnd, GW _ HWND-
FIRST)
'循环直到没有更多的窗口
Do Until mhwnd = 0
'检查是否存在父窗口
If GetParent(mhwnd) = 0 Then
' 检查是否为该进程的窗口
lRes = GetWindowThreadProcessId(mhwnd, mPro-
cID)
If mProcID = ProcessID Then
' 发现了进程的窗口句柄
GetWindowHandle = mhwnd
```

```
    Exit Do
    End If
    End If
    mhwnd = GetWindow(mhwnd, GW_HWNDNEXT)
    Loop
    End Function
```

3. 结束语

由于EPM采用构件化的思想实现，把复杂的API函数调用封装起来，实现了一个简单的编程接口，因此可以很方便地重用。我们已经在多个项目中使用并取得了较好地效果。

参考文献

[1] 曾志民，张秀茂等编，Visual Basic for Windows程序员Windows API参考手册，北京：科学出版社，1995

[2] HOWTO: 32 - Bit App Can Determine When a Shelled Process Ends, Article ID: Q129796 , MSDN, Knowledge Base

[3] How to Find a Window Handle Based on an Instance Handle, Article ID: Q127030 , MSDN, Knowledge Base

[4] Rod Stephens著，钟显宏，李林山，虞里平译，Visual Basic高级技术，北京：电子工业出版社，1998

（来稿时间：1999年5月）