

限制 Windows 应用程序运行实例数

韩振宇 (清华大学自动化系)

Windows 是个多任务操作系统,它不仅可以同时运行多个应用程序,而且可以同时运行一个应用程序的多个实例。但是,有时用户无需或者不能同时运行同一应用程序的多个实例,这时就要限制 Windows 应用程序运行的实例个数。Windows 下的许多应用程序都是只允许一个实例运行的,当用户再次运行该软件时,应用程序自动切换至其第一个实例。

贵刊 1995 年第 8 期刊登的“限制 Windows 应用程序多个实例的方法”中,该文作者提供了一种编程方法,确实可以起到限制一个应用程序只能有一个实例运行的作用,但其中存在一个缺陷。以一个带 About 对话框的应用程序为例,运行此应用程序,通过操作弹出 About 对话框。此时,主窗口为非激活态(标题栏为白色),About 对话框为激活态(标题栏为蓝色)。这时,再次运行此应用程序,如果程序采用了限制实例技术,系统应该恢复运行前一实例,弹出其主窗口和 About 对话框,并且保持主窗口为非激活态,About 对话框为激活态。但是,采用原作者提供的方法编制的程序,实际运行时虽然确实恢复了第一个实例,但窗口的激活状态却改变了,主窗口变为激活态(标题栏为蓝色),About 对话框变为非激活态(标题栏为白色),该“限制 Windows 应用程序多个实例的方法”存在缺陷。

笔者现将“限制 Windows 应用程序运行实例数”的一般编程方法总结如下,供大家参考。

一、总体结构

要限制一个应用程序只能有一个实例运行,首先要在 WinMain() 函数入口判断当前运行的是否是第一个实例。若不是,则不再继续运行,而是激活第一个实例。

程序结构如下:

```
if (hInstPrevious)
{
    激活第一个实例
    return 0;
}
```

运行当前实例;

二、激活第一个实例步骤

1. 获得第一个实例的窗口句柄

方法一:使用 FindWindow() 函数。FindWindow 函数检索

窗口的句柄,此窗口为一主窗口,其类名和窗口名匹配指定的字符串。例如,应用程序的窗口类名为“WndClass”,下面语句将返回第一个实例的窗口句柄。

```
hWndPre = FindWindow("WndClass", 0);
```

方法二:使用 GetInstanceData 函数。GetInstanceData 函数把应用程序先前实例中的数据拷贝到当前实例的数据区。下面语句将从前一实例的数据区拷贝其主窗口句柄。

```
GetInstanceData (hInstPrevious,
                (BYTE *)&hWnd, 2);
```

2. 如果第一个实例的窗口已变为图标,则要在原位置恢复其到原大小

(1) 判断条件

```
if (IsIconic(hWndPre))
```

```
{
    恢复 hWndPre 窗口到原大小;
}
```

(2) 恢复窗口方法

方法一:使用 SendMessage() 函数。SendMessage 函数将指定的消息发送到指定的或所有的窗口。下面语句将向前一实例的窗口发送消息,相当于在系统菜单中选取“还原”命令。

```
SendMessage(hWndPre,
            WM_SYSCOMMAND,
            SC_RESTORE, 0);
```

方法二:使用 ShowWindow() 函数。ShowWindow 函数可设定指定窗口的可见状态。下面语句将恢复并显示第一个实例的窗口。

```
ShowWindow(hWndPre, SW_RESTORE);
```

方法三:使用 OpenIcon() 函数。OpenIcon 函数激活并显示一图标窗口,Windows 将该窗口恢复到原尺寸和位置。下面语句将在原位置按原大小显示前一实例的窗口。

```
OpenIcon(hWndPre);
```

3. 将原实例最近活跃的弹出式窗口或子窗口放到重叠窗口栈的顶部,并激活

(1) 确定第一个实例的最近活跃窗口。使用 GetLastActivePopup() 函数可以确定指定窗口所拥有的弹出式窗口中哪一个最近活跃的。下面语句寻找前一实例的最近活跃窗口。

(2) 将第一个实例的最近活跃窗口放到重叠窗口栈的顶部,并激活。

方法一:使用 BringWindowToTop() 函数。BringWindowToTop 函数将一个弹出式窗口或子窗口放在重叠窗口栈的顶部,同时还激活该弹出式窗口或子窗口。下面语句激活前一实例的最近活跃窗口。

```
BringWindowToTop(hWnd);
```

方法二:使用 SetWindowPos() 函数。SetWindowPos 函数

可改变弹出式窗口、子窗口和顶层窗口的大小、位置和 Z 顺序。下面语句将前一实例的最近活跃窗口放到窗口栈的顶部,并在原位置按原大小显示。

```
SetWindowPos(hWnd, HWND_TOP,
             0, 0, 0, 0,
             SWP_SHOWWINDOW |
             SWP_NOMOVE |
             SWP_NOSIZE);
```

方法三:使用 SetActiveWindow() 函数。SetActiveWindow 函数使指定的顶层窗口成为活跃窗口。下面语句将激活前一实例的最近活跃窗口。

```
SetActiveWindow(hWndPopup);
```

以上三步只是原理上的说明,具体操作时用户可灵活使用。Windows 的“扫雷”程序、“纸牌”程序、“时钟”程序等应用程序都是采用与上面类似的方法限制只有一个实例运行的。需要说明的是,第三步“激活原实例最近活跃的弹出式窗口或子窗口”不能丢,“限制 WINDOWS 应用程序多个实例的方法”就是因为少了此步而造成运行异常。如果用户注意观察的话,会发现 Windows 的“记录器”程序和“剪贴板查看”程序也有类似的异常现象。

作为“限制 WINDOWS 应用程序运行实例数”的一般方法,本文给出了程序一和程序二两种常用形式,供用户使用时参考。该两程序使用 Borland C++ 3.1,在小模式下编译连接,在中西文 Windows 3.1 上运行成功。

程序一:

```
int PASCAL WinMain(HINSTANCE hinstCurrent,
                  HINSTANCE hinstPrevious,
                  LPSTR lpszCmdLine,
                  int nCmdShow)
{
    MSG msg;
    HWND hWndPre;
    if (hinstPrevious)
    {
        hWndPre = FindWindow("WndClass", 0);
        if (IsIconic(hWndPre))
            SendMessage(hWndPre,
                       WM_SYSCOMMAND,
                       SC_RESTORE, 0)

        hWndPre = GetLastActivePopup(hWndPre);
        BringWindowToTop(hWndPre);
        return 0;
    }
}
```

```
(! InitApplication(hinstCurrent))
return FALSE;
if(! InitInstance(hinstZCurrent, nCmdShow))
return FALSE;
while(GetMessage(&msg, NULL, NULL, NULL))
{
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
return(msg.wParam);
```

程序二:

```
HWND hWnd;
int PASCAL WinMain(HINSTANCE hinstCurrent,
                  HINSTANCE hinstPrevious,
                  LPSTR lpszCmdLine,
                  int nCmdShow)
{
    MSG msg;
    HWND hWndPopup;
    if (hinstPrevious)
    {
        GetInstanceData(hinstPrevious,
                       (BYTE *)&hWnd, 2);
        hWndPopup = GetLastActivePopup(hWnd);
        SetWindowPos(hWnd, HWND_TOP,
                    0, 0, 0, 0,
                    SWP_SHOWWINDOW |
                    SWP_NOMOVE |
                    SWP_NOSIZE);
        if (IsIconic(hWnd))
            ShowWindow(hWnd, SW_RESTORE);
        else SetActiveWindow(hWndPopup);
        return 0;
    }
    if (! InitApplication(hinstCurrent))
        return FALSE;
    if (! InitInstance(hinstCurrent, nCmdShow))
        return FALSE;
    while (GetMessage(&msg, NULL, NULL, NULL))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return (msg.wParam);
}
```