Windows 环境下运行外部应用程序的控制

屈景辉　（总后青藏兵站部）

摘要：本文以 *Windows* 环境下的应用系统运行外部程序 *Write* 为背景，叙述如何控制外部应用程序的方法，并给出主要程序片段。

一、问题的引入

在一个决策咨询系统中，需向用户提供格式化文本、图形、图象输出，达到所见即所得的视觉效果。这个问题无论在 DOS 还是 Windows 环境下都是可以解决的。相对而言，在 DOS 环境下如果要达到金山卡文本、图像混排模拟显示效果，则颇费周折，自行设计程序工作量大，周期长，若将金山字处理系统模拟显示功能直接引入应用系统又很麻烦，且显系统笨拙、庞大。而在 Windows 环境下，借助其字处理程序 Write 实现上述目的则不费吹灰之力。Windows 的 Write 不仅使格式化文本具有所见即所得的风格，同时也可实现图文混排，而且操作方便，应用系统也容易控制。

这样，应用系统面临的是如何运行外部程序和对其进行控制的问题。

二、两种运行环境控制分析

一般来说，各种程序设计语言为运行外部程序提供了相应的函数或指令，只要按其要求设置合适的参数即可正常运行。

在 DOS 环境下，应用系统进入外部程序以后，系统资源使用和程序运行控制权也交给外部程序，只有退出外部程序后，系统才受应用系统控制，在两者之间是不能相互切换同时运行的。

由于 Windows 是一个多任务的窗口环境，具有资源共享的特点，它可以同时运行多个任务或一个任务的多个实例，任务与任务之间、实例与实例之间可以快速切换，也就是说，在应用系统启动以后，用户可以切换到其它程序运行，或当应用系统运行外部程序时，该程序（任务）窗口很可能覆盖应用系统窗口，颇显喧宾夺主，使用户失去约束，使当前任务窗口成为非激活窗口而被激活窗口覆盖，使缺乏经验用户束手无策。因此，在设计系统时要调用外部程序应采取必要措施，禁止与应用系统无关的任务运行，使外部程序始终处于应用系统的控制之中。

三、控制策略

为叙述方便，以实现具有格式化文本、图文混排文档资料的显示为目的，调用 Windows 的 Write 程序说明控制的实现策略。

1.应用系统窗口外观控制

在建立系统窗口时，设定窗口具有系统菜单、标题栏风格，且窗口大小充满整个屏幕，显示时设定参数为 SW__SHOWMAXIMIZED，使窗口激活以最大方式显示。以防止用户随意掀动鼠标按钮而切换任务窗口。

2.运行任务控制

(1)任务的启动与关闭。在应用系统中，启动外部程序可用下列两个函数

```
UINT WinExec(LPCSTR lpszCmdLine,UINT fuCmdshow);
HINSTANCE LoadModule(LPCSTR lpMdNm,LPVOID lpPara-
Blk);
```

装载并执行指定的 Windows 应用程序。函数

```
void TerminateApp(HTASK htask,WORD wFlags);
```

关闭指定的应用程序实例。

(2)运行任务的搜索。搜索运行任务的方法有两种：

方法 1:查找指定任务窗口。任何一个任务都是以窗口或图标形式运行，用函数

```
HWND FindWindow(LPSTR lpszClassName,LPSTR lpsz-
Window);
```

查找与指定类名、窗口名匹配的窗口句柄，根据返回值确定某一任务是否运行。例如：若目前任务队列中没有指定任务在运行，便启动该任务，可用下列条件语句实现：

```
If (FindWindow(NULL,"窗口标题") = = NULL)
    WinExec("任务模块名",显示方式);
```

方法 2:搜索任务队列。Windows 提供了三个函数来

实现任务队列搜索:

```
BOOL TaskFirst(TASKENTRY FAR * lpte);
BOOL TaskNext(TASKENTRY FAR * lpte);
BOOL TaskFindHandle(TASKENTRY FAR * lpte,HTASK htask);
```

函数 1、3 搜索第一任务和指定任务信息,函数 2 搜索任务队列后续条目。应用程序根据函数填充任务队列结构 TASKENTRY 的信息判定运行任务情况。从结构成员 szModule[]、hTask 内容可知运行任务模块名和任务句柄。

下列程序段搜索任务队列中运行的 Write 实例的数目并将其关闭:

```
BOOL nTask; TASKENTRY teTask; //定义任务队列结构
int nTaskNum = 0; teTask.dwSize = sizeof(TASKENTRY);
nTask = TaskFirst(teTask);
while (nTask != 0) {
    if(!lstrcmp(teTask.szModule,"WRITE")) {
    TerminateApp(teTask.hTask,NO UAE BOX);
    nTaskNum++; }
nTask = TaskNext(teTask); }
```

若将 if 语句的条件和函数改变,还可将任务队列中与应用系统无关的任务关闭,或者根据任务队列是否有指定任务来确定其是否启动。

3.运行任务窗口位置控制

(1)获取任务窗口句柄。若在对话框编辑控制子窗口内用 Write 显示图文混排文档 filename.txt,让 Write 充满子窗口。用函数

```
WinExec("WRITE FILENAME.TXT",SW HIDE);
```

将其显示方式设置为 SW_HIDE 暂让 Write 窗口隐藏,否则就随意显示,若任务已在运行可省去这一步。然后获取 Write 窗口句柄:

```
hwnd = FindWindow(NULL,"WRITE - FILENAME.TXT");
```

(2)获取显示区域矩形坐标。用函数

```
GetWindowRect(hCtrlEdit,&rc);
```

获得编辑控制子窗口区域坐标存入矩形结构 rc 内。

(3)定位显示。若已知窗口在非极小化状态时,根据结构 rc 的值用

```
SetWindowPos(hwnd,HWND TOP,rc.left,rc.top,rc.right,
    rc.bottom,SWP SHOWWINDOW);
```

函数将 Write 窗口放置在编辑控制子窗口内。注意该函数的第二、七个参数设置为 HWND_TOP、SWP_SHOWWINDOW,使窗口按指定区域放在 Z 顺序顶部正常显示。

当窗口处于极小化状态时初始化 WINDOWPLACEMENT 位置数据结构并显示:

```
WINDOWPLACEMENT lpwndpl;
lpwndpl.length = sizeof(WINDOWPLACEMENT);
lpwndpl.flags = WPF RESTORETOMAXIMIZED;
lpwndpl.showCmd = SW RESTORE;
lpwndpl.rcNormalPosition = rc;  //获取的矩形区域坐标
SetWindowPlacement(hwnd, lpwndpl);
```

为使程序具有通用性,无论任务在什么状态运行,首先用函数 CloseWindow(hwnd)将其极小化,统一采用窗口极小化时的定位显示方法处理。

四、一个控制 Write 运行的例子

下面是在对话框编辑控制子窗口控制 Write 启动、显示、关闭的对话框过程程序段,且保证需要显示时只有一个 Write 在运行,退出对话框时将其关闭。

```
:
(有关结构、变量初始化同前)
...
case IDD DISP:    //在编辑子窗口用 Write 显示文本
nTask = TaskFirst(teTask);    //搜索任务队列
  while (nTask != 0) {
    if(!lstrcmp(teTask.szModule,"WRITE")) {
      nTaskNum++;
      break; }
    nTask = TaskNext(&teTask); }
  if(nTaskNum != 0) {    //有 Write 运行的处理
  GetWindowRect(hCtrlEdit,&rc);
  hwnd = Findwindow(NULL,"窗口标题");
  CloseWindow(hwnd);
  SetWindowPlacement(hwnd,&lpwndpl); }
  else {    //无 Write 运行的处理
    WinExec("WRITE.EXE",SW HIDE);
    GetWindowRect(hCtrlEdit,&rc);
    hwnd = FindWindow(NULL,"窗口标题");
    SetWindowPos(hwnd,HWND-TOP,rc.left, rc. top,
        rc.right,rc.bottom,SWP-SHOWWINDOW);
    }
  return TRUE;
case IDD'OK:    //返回
    nTask = TaskFirst(&teTask); //任务队列搜索
    while (nTask != 0) {    //关闭 Write.EXE 的运行
      if (!lstrcmp(teTsak.szModule,"WRITE"))
        TerminateApp(teTask.hTask, NO UAE BOX);
  nTask = TaskNext(teTask); }
    EndDialog(hDispDlg,TRUE);
    return TRUE;
```

参考文献:

[1]王旭等译,《Micosoft Windows 3.1 程序员参考大全(二)---函数》,清华大学出版社,1993,7