控制 Win32 应用执行的方法

王 东 林冬梅 唐国维 (大庆石油学院计算机科学系 151400)

摘要:本文简要分析了 Win32 环境中的应用程序共享方式,针对没有源代码及访问接口函数的应用程序的集成,给出了一种利用消息和线程实现应用程序控制执行的方法,分析了该方法实现过程中存在的典型问题,并提出了相应问题的解决方法。

关键词:Win32 控制执行 消息 线程

1. 问题的引出

在现有的 Windows 环境中,已经存在大量实用工具和应用程序。这些宝贵的程序资源,主要以公共对象控制(COM)、动态连接库(DLL)形式或可执行应用程序(Executable)形式表现。对于第一种情况,应用是以控件(OCX)形式表现,而现有的 Windows 32 位开发环境均提供了对 OCX 控件的直接编程使用,例如 Visual C++、Delphi、Power Builder 等。对于第二种情况,如果提供了应用程序接口(API),程序设计人员仍可以在程序开发或发布中直接使用,例如 Windows 环境中所有开放的程序资源,均提供了形式化 API 函数访问接口描述;相反的情况下则不然,由于没有形式化 API 函数访问接口描述,程序设计人员很难使用其内部功能。对第三种情况,如果应用程序提供了动态数据交换(DDE)和对象连接嵌入技术(OLE),则程序设计人员仍然可以在程序中控制和使用该应用程序;但很多应用程序不提供这些技术,因此必须通过其他办法来控制应用程序的执行。本文提出了一种利用消息和线程解决上述问题的方法。

2. 控制方法

由于 Windows 环境中,应用程序执行的各种控制,多数是由操作人员通过输入设备形成输入,Windows 系统捕获输入后,形成消息发送给当前应用的主程序,再由这个主程序将消息派送到该应用程序的当前窗口。所以,上述控制应用程序方法的基本思路是:利用 Windows 环境中的消息机制实现应用程序执行流程的控制。具体的说,是在待研制的应用程序中,利用 Windows 的 API 函数向被控制的应用程序发送其能够正确解释的消息,从而达到控制该应用程序执行过程的目的。在这种方法中,应注意以下几点:

第一,正确获得当前被控制的类名。在 Windows 环境中,无论任何用户界面性组件,均是以类的形式在系统中进行注册并分配一个唯一标识——句柄(handle)。系统在运行过程中,常规的应用程序类是比较好捕获并确定的,而对于诸如对话框(Dialog)这样的窗口,在不同的系统中存在细微的差别,其注册的类名是以代号形式表现的。如果不能正确获得当前被控制的类名,则不能获得该类的句柄,进而不能对该类进行消息级的控制。

第二,形成的消息队列必须能够正确实现被控制应用程序的执行流程。一般来说,被控制的应用程序在控制执行过程中所完成的操作,与由操作人员直接用输入设备控制其完成的操作具有相同的视觉效果。所以设计的消息队列,与操作人员控制被控制应用程序执行过程中,由 Windows 发送的消息队列相同,才能达到相同的效果。

第三,消息的发送必须与该消息所控制的对象对应。在 Windows 环境中,尤其是在 32 位的环境中,Windows 系统是采用抢先式调度策略对应用进行管理的。所以,在利用消息对某个应用控制执行过程中存在这种情况,该被控制应用创建了一个新的窗口,新窗口的创建是需要占用较长的 CPU 时间的,而控制程序的执行是不等待被控制应用程序的执行。这样就会导致控制程序发送给新窗口的消息,由于新窗口没有准备好被丢失。如果采用查询式程序设计方法则会遇到系统没有任何响应的问题,其原因是控制程序进入循环状态,占用了绝对的 CPU 的时间,系统不能创建新的窗口而导致操作系统进入"busy"状态。

如何解决上述问题是本技术实现的关键所在。

3. 消息和线程

应用程序所要做的每项工作几乎都是基于 Windows 消息的,这些消息分 3 种基本形式:常用的 Windows 消息、控制通知和命令。常用的 Windows 消息,其标识加上前缀 WM_,代表发生在应用程序中窗口处理消息。

控制通知消息是由子窗口传向主窗口的 WM _ COMMAND 消息。命令是由菜单、按钮和快捷键传递来的 WM _ COMMAND 消息。在 Windows 环境中,应用程序可以通过调用其 API 函数实现消息的发送。

Windows 的 32 位操作系统支持多进程技术,同时每个进程可以进一步形成若干线程。任何一个应用程序或进程都有唯一的原线程,并可以启动其他线程。所有线程共享创建它们的进程的内存空间。当一个程序拥有多个线程时,线程的执行顺序是随机的。所以线程的设计必须考虑资源共享的问题。线程大体上分为工作线程(无用户界面)和用户界面线程两种类型。对于 Windows 环境中的消息和线程技术更详细的内容请参考有关资料。

4.关键技术实现

下面针对实现对应用程序控制的几个关键的技术进行深入的讨论。

(1)捕获控制类名。前面已经讲到,在应用程序运行过程中,所有与用户界面有关的组件通常是 Windows 环境中某个标准的类。当其以图形方式表示在输出设备上时,其所对应的类已经在 Windows 系统内部注册,如窗口、标签、菜单、输入框、树形结构、列表框、对话框等。通常情况下,这些组件的类名是固定的,但随操作系统的不同,其注册的类名也不相同。所以在进行上述工作之前,必须利用类似于 Win Sight 的工具,正确获得需要控制对象的类名。如在 Windows 98 环境中,打印机对话框、打印机属性对话框的注册类名是"#32770",而不是通常的 TDialog 等。

(2)建立控制消息队列。控制消息队列的建立,需要对 Windows 环境中的消息机制有深入的理解。在此基础上,按照对应用程序控制流程,形成操作历史记录;然后根据历史记录,将操作分解成对应的消息,进而形成消息队列。其中有一点需要注意的是,由于操作过程中,操作人员可能对应用程序中的不同窗口操作以完成最终的任务,所以在形成历史记录过程中,应该详细记录每个操作是属于哪个窗口的,这样在形成消息时可以将消息分派到不同的窗口中,以能够形成正确的控制。相反情况下,如果本来按照窗口 A 操作过程形成的消息,被发送到窗口 B,则窗口 B 可能不能按照预定的操作过程完成相应的操作。

5.过程实现

下面以控制 Windows 环境中的打印机窗口为例介绍控制具体的实现过程。在这个例子中,实现选择自定义纸张,并设定纸张的长度、宽度以及边距,打印机类型为 LQ - 1600K,实现环境是 Microsoft Windows 98 以及 Borland Delphi V3.0。

(1)捕获窗口句柄。得到某个控制对象的类名之后,在待开发的应用程序中,可以利用 Windows API 函数获得该对象的实例化句柄,只有得到该句柄后才可以进一步对该对象控制。下面的程序是获得"Epson LQ - 1600K"窗口句柄的样例。

```
myHWND := FindWindow( '#32770', 'Epson LQ - 1600K' );
while myHWND = 0 do
    myHWND := FindWindow( '#32770', 'Epson LQ - 1600K' );
```

(2)向窗口发送消息。获得某个对象的句柄后,就可以向该对象发送其可以接受的各种消息。下面是选择打印机窗口的"打印机"菜单下的"属性"子菜单程序样例。

```
SendMessage( myHWND, WM-IME-NOTIFY, 2, 0 );
SendMessage ( myHWND, WM-MENUSELECT, $00900CE0, $00000CE4 );
SendMessage ( myHWND, WM-MENUSELECT, $008000A2, $00000CE0 );
SendMessage ( myHWND, WM-MENUSELECT, $00800073, $000009D8 );
SendMessage ( myHWND, WM-MENUSELECT, $FFFF0000, $000000000 );
SendMessage ( myHWND, WM-COMMAND, $00000073, $00000000 );
SendMessage( myHWND, WM-IME-NOTIFY, 1, 0 );
```

(3)为待创建窗口生成等待进程。在一个应用程序运行过程中,可能需要设计许多窗口配合完成特定的任务。所以在控制应用程序运行过程中,可能遇到等待窗口的问题。等待窗口的原因在前面已经进行了分析,并提出的方法是利用线程来解决这种问题。许多 Win32 开发环境中均支持线程技术。这里有一点需要说明的是,由于线程具备与进程相似的特点,所以在使用线程时,必须保证线程是可结束的。第一,线程程序本身必须有结束条件;第二,线程的结束条件必须出现,否则这个线程将变成"死"线程,产生不可预测的后果。

下面的程序演示了当启动设置打印机自定义纸张页长和页宽窗口,并将页长和页宽设置为某个固定值。

```
    { 启动设置打印机自定义纸张页长和页宽线程,该线程等待下面程序段启动"自定义尺寸""窗口 }
    SetupHB.Create( PageLength, PageWidth );
    { 选择自定义纸张程序段,启动"自定义尺寸"窗口 }

    {
    { * * * * 线程部分 * * * * }
    { 查找"用户自定义尺寸"窗口句柄 }
    { 查找"用户自定义尺寸"窗口中纸张宽度 }
    { 设置"用户自定义尺寸"窗口中纸张宽度 }
    { 查找"用户自定义尺寸"窗口中纸张长度 }
    { 设置"用户自定义尺寸"窗口中纸张长度 }
    { 关闭窗口 }
    { 结束 }
```

5. 结束语

上述 Win32 环境下利用消息和线程对应用程序控制的方法,可以进一步应用到演示系统的制作、实用程序或工具的执行控制、多种应用程序的集成等。可以按照实际需求,对上述方法进行改进和扩充,以实现更复杂、实用的技术。

参考文献

[1] Mattbew Tells, Andrew Cooke 著, Windows 95 API 开发人员指南,机械工业出版社

[2] Borland 公司, Delphi 编程指南,石油工业出版社

(来稿时间:1999 年 6 月)