# Windows3.1 编程入门系列讲座(下)

## 第三讲 深入理解 windows3.1 程序设计风格

李晓华　(云南省军区自动化站　650051)

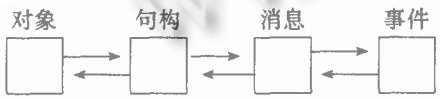
### 一、Windows 的消息循环

windows 只能通过事件给应用程序发送信息,或者从应用程序接收信息。这些信息就称为消息。

windows 中的消息具有二层意思。

其一:windows 要处理含有很多预定义的消息资源集,这是消息与公共事件紧密相联。

其二:这些消息是与消息所影响的对象相联系的,从而我们得出以下这些关系:




一旦创建显示一个窗口后,windows 函数可以开始行使它的主要职责,从应用程序队列中读取消息并经它们发送到对应的窗口,在 windows 中通过一个 while()消息循环来实现的。在 windows 中通常有以下几种消息循环方式:

**1.最简单的消息循环**

最简单的消息循环由两个函数 GetMessage 和 DispatchMessage 组成。其格式如下:

```
MSG msg;
....
while(getMessage(&msg, NULL, NULL, NULL)){
    DispatchMessage(&msg);
}
```

其中函数 GetMessage 从应用程序队列中检索消息并将其拷贝到"msg"的消息结构中去。函数 DispatchMessage 指示 Windows 将每条消息发送给相应的窗口函数。

**2.能处理键盘输入的消息循环**

为了处理键盘的字符输入,必须使用函数 TranslateMessage 来翻译消息,其格式如下:

```
while(getMessage(&msg, NULL, NULL, NULL)){
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
```

**3.具有加速表的消息循环**

要处理加速表,必须在消息循环中加入函数 TranslateAccelerator 函数。其格式如下:

```
while(getMessage(&msg, NULL, NULL, NULL){
if (! TranslateAccelerator(hWnd, hAccTable, &msg))
    {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
    }
}
```

其中 IF 语句检查每个消息,看它是否为一加速键消息。hWnd 标志窗口以表明需要翻译那一个窗口的消息。如不是加速键消息,应用程序象通常那样处理该消息。

**4.拥有非模式会话框中的消息循环**

```
while(getMessage(&msg, NULL, NULL, NULL){
  if(hDlg = NULL||! isdialogMessage(hDlg, &msg))
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
```

其中 isDialogMessage 函数决定该消息是否为会话框的,如是,执行消息,但不再由 TranslateMessage(&msg),DispatchMessage(&msg)函数作进一步的处理。

**5.MDI 的消息循环**

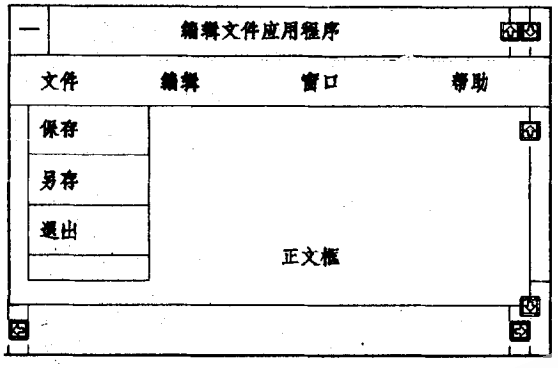
MDI 和非 MDI 应用程序的主消息循环是非常相似的。不同的是 MDI 应用程序用函数 TranslateMDISysaccel()来翻译子窗口的加速键。

```
while(getMessage(&msg, NULL, NULL, NULL){
 if(hDlgModeless! = NULL&IsDialogMessage(hDlgModeless, &msg))
    &&! TranslateMDISysAccel(ghwndlient, &msg)
    &&! TranslateAccelerator(ghwndFrame, hAccelTable, &msg)
    )
    {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
    }
}
```

### 二、创建 Windows 应用程序窗口

窗口是 windos 应用程序的主要输入手段,是应用程序存取系统显示器的唯一方法。一个典型的 windows 窗口见下图所示,它具有标题栏、主菜单、流动杆、边框及在 CRT 上占用一个矩形区域的特征,用户在使用窗口前必须创建窗

口。




在创建窗口前，必须先注册窗口，其注册窗口必须填写WNDCLASS结构，见常用结构，并使用 RegisterClass 向窗口注册。向窗口注册主要是定义光标形状、菜单字符以及该窗口的消息事件。

创建一窗口通常应包括以下几个步骤：

1.建立一个指向类型 WNDCLASS 结构指针，主要包括窗口类的名字、属性、资源、窗口函数等。

使用 GreateWindows 来创建窗口，其格式如下：

```
hWnd = CreateWindow (
    lpClassName,        窗口类型名
    lpWindowsName,      窗口标题
    dwStyle,            窗口风格
    X,                  窗口位置
    Y,                  窗口位置
    nWidth,             窗口宽度
    nHeight,            窗口高度
    hWndParent,         父窗口句柄
    hMenu,              菜单句柄
    hInstance,          事例句柄
    lpParam             附加信息
    );
```

2.根据结构 WNDCLASS 的大小，用 LocalAlloc()分配局部空间。

3.用 Windows 的函数 RegisterClass()将结构传给 Windows。

一旦完成上述，则就可以用 Showwindows、Updatewindows 来显示或更改窗口的用户。其格式如下：

showWindows (hwnd, nCmdShow)显示窗口。

UpdateWinodws(haw) 更改窗口。

下面是窗口函数格式：

```
long FAR PASCAL WndProc ( hWnd, messagee, wParam, lParam)
HWND hWnd;              窗口句柄
unsigned message;       消息的类型
WORD wParam;            有关消息的附加信息
LONG lParam;            有关消息的附加信息
{
    .....
    PAINTSTRUCT ps;
    switch(message);
    {
    case WM-DESTROY;
        psotQuitMessage(0);
        break;
    case WM-PAINT;
        BeginPaint(hWnd,(LPPAINTSTRUCT)&ps);
        EditPaint(ps.hdc);
        EndPaint(hWnd,(LPPAINTSTRUCT)&ps);
        break;
    ..........
    default;
    return DefWindowsProc ( hWnd, message, wParam, lParam);
    break;
    }
}
```

窗口函数总是使用 PASCAL 调用约定。因为 Windows 调用一个函数时总是使用 32 位地址，故窗口定义必须有 FAR。

由于窗口函数从 Windows 接收消息是不相同的，它既有 WinMain 函数中消息循环发来的消息(鼠标输入、键盘输入及时钟输入。典型的信息有：WM-KEYDOWN、WM-KEYUP、WM-MOUSEMOVE、WM-TIME)；又有从 Windows 来的窗口管理消息(如 WM-PAINT、WM-DESTORY、WM-CREATE 等)。所以使用一个 swith(message)来处理不同的消息类型。

## 三、制作 Windows 应用程序的菜单

1.Windows 下菜单的含义

菜单是用户在 windows 应用程序中的主要输入手段。菜单是命令表，用户可以察看、选择其中的项目。windows 环境下的菜单分为下拉式、弹出式、级取式、动态式等菜单。当用户选择一个菜单项，windows 发送 WM－command 消

息。

2.如何制作一个 Windows 应用程序的菜单

制作一个 Windows 下应用程序的菜单是非常简单的。它必须按下列方法进行：

(1)在应用程序资源文件(*.RC)中指定菜单项；

```
theMenu MENU
    Begin
      POPUP "&File"
        Begin
          MENUITEM "&Open", WN-OPEN
          MENUITEM "&New", WN-NEW
          MENUITEM "&Save", WN-SAVE
          MENUITEM "&Exit", WN-EXIT
        END
      POPUP "&Edit"
          Begin
          MENUITEM "&Undo", WN-UNDO
          MENUITEM "&Copy", WN-COPY
          END
      .....
    End
```

其中菜单命名为 theMenu，MENU 为关键字，由一对 Begin 和 End 包括了菜单的菜单项.每个菜单项有一个唯一的标识符 WN-OPEN、WN-NEW..，通常称"菜单 ID"。"菜单 ID"必须在资源文件中定义为一常量：

```
#define WN-OPEN 100
#define WN-NEW 110
.....
```

(2)在应用程序源码(*.C)中指定菜单。有两种方法：

①在注册窗口时，对于整个窗口类指定一个菜单。如：

```
wc.lpsMenuName = "theMenu";
```

lpsMenuName 域名是 WC 的数据结构 WNDCLASS 的部分。theMenu 是在资源文件中给定的菜单。

②创建窗口时，为该窗口指定一个菜单。如：

a.使用 LoadMenu 从应用程序资源中装载菜单。

```
hWnd = CreateWindow (
    lpClassName,        窗口类型名
    lpWindowsName,      窗口标题
    dwStyle,            窗口风格
    X,                  窗口位置
    Y,                  窗口位置
    nWidth,             窗口宽度
    nHeight,            窗口高度
    hWndParent,         父窗口句柄
    hMenu = LoadMenu (hInstanec, (LPSTR)" theM-
    menu")  菜单句柄
    hInstance,          事例句柄
    lpParam             附加信息
    );
```

(3)根据需要可初始化菜单；

(4)处理从菜单输入的消息：一旦用户选择了菜单中的一项，Windows 将消息 WN-COMMAND 送给相应的窗口函数。其该"菜单 ID"包含在消息的参数 wParam 中，它是通过一个 SWITCH 语句不同功能的选择：

```
case WN-COMMAND
  switch(wParam){
    case WN-OPEN
      ..... 执行打开文件的操作
      break;
    case WN-NEW
      ..... 执行一个新文件的操作
      break;

  break;
```

3.与菜单有关的函数

一个 Windows 的菜单项，可以根据需要对它进行一系列处理，如使菜单项变暗、有效、无效、检查、增加、改变、删除、使用图位等作操作。

(1)打开、关掉菜单函数：使用 Windows 的 EnableMenu-Item()函数，可以使一个菜单项变灰或无效。

其格式如下：

BOOL EnableMenuItem(hMenu, wIDItem, wEnable)

其中：hMenu 为菜单句柄

wIDItem 为菜单项的 ID

wEnable 为可选项：

| | |
|---|---|
| MF-GRAYED | 使菜单变灰 |
| MF-ENABLED | 使菜单有效 |
| MF-DISABLES | 使菜单无效 |
| MF-BYPOSITION | 对应菜单项的位置 |
| MF-BYCOMMAND | 资源菜单 ID |

(2)检查菜单函数：BOOL CheckmenuItem (hMenu, wIDtem, wCheck)

其中：hMenu 为菜单句柄

wIDItem 为菜单项的 ID

wCheck 为可选项：

MF-CHECKED 检查菜单项

MF-UNCHEKED 去掉菜单项检查

MF-BYPOSITION 对应菜单项的位置

MF-BYCOMMAND 资源菜单 ID

(3)增加菜单项函数

①在尾部添加一个菜单项。BOOL AppendMenu (hMenu, wEnable, aMenuItem, string)

其中：hMenu 为菜单句柄

wEnable 为可选项：指明该项是检查的、打开的、灰色的等。

aMenuItem 为添加一项菜单。

string 为菜单项的名字。

②在指定位置播入一个菜单项。BOOL InsertMenu (hMenu, oldMenuItem, wEnable, aMenuItem, string)

其中：hMenu 为菜单句柄

oldMenuitem 为指定的菜单项。

wEnable 为可选项：指明该项是检查的、打开的、灰色的等。

aMenuItem 为添加一项菜单。

string 为菜单项的名字。

(4)修改已存在的菜单项：BOOL modifyMenu (hMenu, oldMenuItem, wEnable, nMenuItem, "string")

其中：hMenu 为菜单句柄

oldMenuitem 为指定要修改的菜单项。

wEnable 为属性。

nMenuItem 为被改成的菜单项。

string 为菜单项的名字。

(5)删除一个菜单项：BOOL DeleteMenu(hMenu, nPosition, wFlags)

其中：hMenu 为菜单句柄

nPosition 要删除的项目的一个数字

wFlags 包含下列选项：

MF-BYPOSITION nPosition 基于菜单项的位置

MF-BYCOMMAND nPosition 包含一个资源 ID 号

(6)使用图位作菜单有两种方法

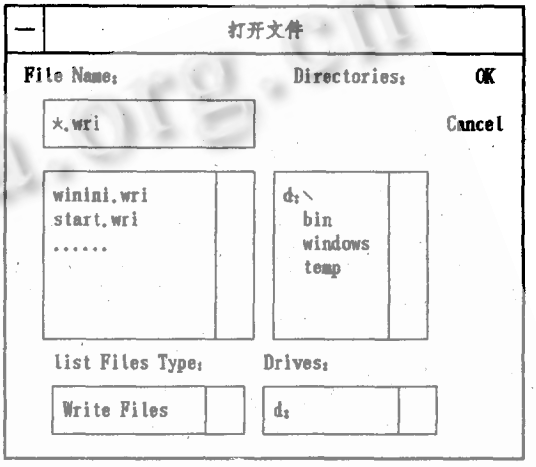
①当插入或附加一个新的菜单项时，指定按位图显示该菜单项。

②用函数 ModifyMenu 修改一个存在的菜单项，让它按位图显示。

## 四、建立对话框

对话框是一种弹出式窗口以供应用程序和用户之间进行交互。通常，对话框是被用来为用户提示信息以便用户完成各种操作或动作。一个对话框拥有一个或多个的控制。以备用户进入文本文件。windows 有两种不同的对话框：模式对话框和非模式对话框。

### 1. 模式对话框




模式对话框要求用户在应用程序继续工作之前必须作出响应，也就是说，它必须暂时关闭窗口并且在返回控制父窗口之前强迫用户来完成所要求的操作。模式对话框对于为了程序运行而收集应用程序的信息有特别用处。例如当用户从 File 菜单中选择 open 命令时，windows 则显示下列的对话框：这样只有当处理完 open 这个对话框之后，才能返回 windows 应用环境。

通过调用 DialogBox()函数来显示模式对话框，通过调用 EndDialog()函数来关闭模式会话框。

在模式会话框中，可能用 EnableWindows(hWnd, bEnable)函数来使某些选按纽变暗，表示它处于非活动状态。它一般用来允许或禁止一个窗口。

### 2. 非模式对话框

非模式对话框在处于活动状态时仍允许用户在应用程序中继续工作。例如 windows 的 write 为自身的 Find 命令而使用非对话框，这就使得在不关闭 Find 对话框时允许用户继续编辑文件。

可以使用 Great Dialog()函数来建立非模式对话框。在使用非模式对话框时，应用程序的主 Getmessage() - DispatchMeisapt 循环仍然接受消息，而模式对话框有自己的 Getmessage() - DispatchMessage()消息处理循环，并通过消息循环中的 IsDialogMessage()命令直接发送到对话框。