Win32 汇编学习 (6)：键盘输入消息

作者：Akkuman
原文链接：https://ld246.com/article/1518196844670
来源网站：链滴

这次，我们将要学习WINDOWS程序是如何处理键盘消息的。

<!--more-->

理论：

因为大多数的PC只有一个键盘，所以所有运行中的WINDOWS程序必须共用它。**WINDOWS 将负把击键消息送到具有输入焦点的那个应用程序中去。**尽管屏幕上可能同时有几个应用程序窗口，但一时刻仅有一个窗口有输入焦点。有输入焦点的那个应用程序的标题条总是高亮度显示的。 实际上您可从两个角度来看键盘消息：一是您可以把它看成是一大堆的按键消息的集合，在这种情况下，当您按一个键时，WINDOWS就会发送一个 WM_KEYDOWN 给有输入焦点的那个应用程序，提醒它有一键被按下。当您释放键时，WINDOWS又会发送一个 WM_KYEUP 消息，告诉有一个键被释放。您每一个键当成是一个按钮；另一种情况是：您可以把键盘看成是字符输入设备。当您按下“a”键时WINDOWS发送一个 WM_CHAR 消息给有输入焦点的应用程序，告诉它“a”键被按下。实际上WIDOWS 内部发送 WM_KEYDOWN 和 WM_KEYUP 消息给有输入焦点的应用程序，而这些消息将通调用 TranslateMessage 翻译成 WM_CHAR 消息。WINDOWS窗口过程函数将决定是否处理所收到消息，一般说来您不大会去处理 WM_KEYDOWN 、 WM_KEYUP 消息，在消息循环中 TranslateMesage 函数会把上述消息转换成 WM_CHAR 消息。这次学习中将只处理 WM_CHAR。

例子：

```
.386
.model flat,stdcall
option casemap:none

WinMain proto :DWORD,:DWORD,:DWORD,:DWORD

include windows.inc
include user32.inc
include kernel32.inc
include gdi32.inc
includelib user32.lib
includelib kernel32.lib
includelib gdi32.lib

.data
ClassName db "SimpleWinClass",0
AppName   db "Our Fourth Window",0
char WPARAM 20h

.data?
hInstance HINSTANCE ?
CommandLine LPSTR ?

.code
start:
invoke GetModuleHandle,NULL
mov    hInstance,eax
invoke GetCommandLine
mov    CommandLine,eax
invoke WinMain,hInstance,NULL,CommandLine,SW_SHOWDEFAULT
invoke ExitProcess,eax
```

```
WinMain proc hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,CmdShow:DWORD

    LOCAL wc:WNDCLASSEX
    LOCAL msg:MSG
    LOCAL hwnd:HWND
    mov wc.cbSize,SIZEOF WNDCLASSEX
    mov wc.style,CS_HREDRAW or CS_VREDRAW
    mov wc.lpfnWndProc,OFFSET WndProc
    mov wc.cbClsExtra,NULL
    mov wc.cbWndExtra,NULL
    push hInst
    pop wc.hInstance
    mov wc.hbrBackground,COLOR_WINDOW+1
    mov wc.lpszMenuName,NULL
    mov wc.lpszClassName,OFFSET ClassName
    invoke LoadIcon,NULL,IDI_APPLICATION
    mov wc.hIcon,eax
    mov wc.hIconSm,eax
    invoke LoadCursor,NULL,IDC_ARROW
    mov wc.hCursor,eax
    invoke RegisterClassEx,ADDR wc
    invoke CreateWindowEx,NULL,ADDR ClassName,ADDR AppName,\
                    WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,CW_USEDEFAULT,\
                    CW_USEDEFAULT,CW_USEDEFAULT,NULL,NULL,hInst,NULL
    mov hwnd,eax
    invoke ShowWindow,hwnd,SW_SHOWNORMAL
    invoke UpdateWindow,hwnd
    .while TRUE
        invoke GetMessage,ADDR msg,NULL,0,0
        .break .if (!eax)
        invoke TranslateMessage,ADDR msg
        invoke DispatchMessage,ADDR msg
    .endw
    mov eax,msg.wParam
    ret

WinMain endp

WndProc proc hWnd:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM

    LOCAL hdc:HDC
    LOCAL ps:PAINTSTRUCT

    .if uMsg==WM_DESTROY
        invoke PostQuitMessage,NULL
    .elseif uMsg==WM_CHAR
        push wParam
        pop  char
        invoke InvalidateRect,hWnd,NULL,TRUE
    .elseif uMsg==WM_PAINT
        invoke BeginPaint,hWnd,ADDR ps
        mov    hdc,eax
        invoke TextOut,hdc,0,0,ADDR char,1
```

```
        invoke EndPaint,hWnd,ADDR ps
    .else
        invoke DefWindowProc,hWnd,uMsg,wParam,lParam
        ret
    .endif
    xor eax,eax
    ret

WndProc endp
end start
```

分析：

```
char WPARAM 20h
```

这个变量将保存从键盘接收到的字符。因为它是在窗口过程中通过WPARAM型变量传送的，所以我简单地把它定义为WPARAM型。**由于我们的窗口在初次刷新时(也即刚被创建的那一次)是没有键盘入的所以我们把他设成空格符（20h），这样显示时您就什么都看不见。**

```
.ELSEIF uMsg==WM_CHAR
    push wParam
    pop  char
    invoke InvalidateRect, hWnd,NULL,TRUE
```

这一段是用来处理WM_CHAR消息的。它把接收到的字符放入变量char中，接着调用InvalidateRect而InvalidateRect使得窗口的客户区无效，这样它会发出WM_PAINT消息，而WM_PAINT消息迫使WNDOWS重新绘制它的客户区。该函数的语法如下：

```
InvalidateRect proto hWnd:HWND, lpRect:DWORD, bErase:DWORD
```

lpRect是指向客户区我们想要其无效的一个正方形结构体的指针。如果该值等于NULL，则整个客户都无效；布尔值bErase告诉WINDOWS是否擦除背景，如果是TRUE，则WINDOWS在调用BeginPaint函数时把背景擦掉。 所以我们此处的做法是：**我们将保存所有有关重绘客户区的数据，然后发送WM PAINT消息(通过InvalidateRect)，处理该消息的程序段然后根据相关数据重新绘制客户区。实际上们完全可以通过调用 GetDC 获得设备上下文句柄，然后绘制字符，然后再调用ReleaseDC释放设备下文句柄，毫无疑问这样也能在客户区绘制出正确的字符。但是如果这之后接收到WM_PAINT消息处理时，客户区会重新刷新，而我们这稍前所绘制的字符就会消失掉。所以为了让字符一直正确地显，就必须把它们放到WM_PAINT的处理过程中处理。而在本消息处理中发送WM_PAINT消息即可。**

```
invoke TextOut,hdc,0,0,ADDR char,1
```

在调用InvalidateRect时，WM_PAINT消息被发送到了WINDOWS窗口处理过程，程序流程转移处理WM_PAINT消息的程序段，然后调用BeginPaint得到设备上下文的句柄，再调用TextOut在客区的（0，0）处输出保存的按键字符。这样无论您按什么键都能在客户区的左上角显示，不仅如此，论您怎么缩放窗口（迫使WINDOWS重新绘制它的客户区），字符都会在正确的地方显示，所以**必须所有重要的绘制动作都放到处理WM_PAINT消息的程序段中去。**