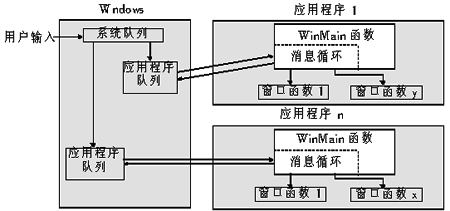


[**windows消息机制**](http://www.cppblog.com/mzty/archive/2006/11/24/15619.html)

一 Windows中有一个系统消息队列，对于每一个正在执行的Windows应用程序,系统为其建立一个“消息队列”，即应用程序队列，用来存放该程序可能创建的各种窗口的消息。应用程序中含有一段称作“消息循环”的代码，用来从消息队列中检索这些消息并把它们分发到相应的窗口函数中。



二 Windows为当前执行的每个Windows程序维护一个「消息队列」。在发生输入事件之后，Windows将事件转换为一个「消息」并将消息放入程序的消息队列中。程序通过执行一块称之为「消息循环」的程序代码从消息队列中取出消息：  
while(GetMessage (&msg, NULL, 0, 0))         
{          
    TranslateMessage (&msg) ;          
    DispatchMessage (&msg) ;         
}

msg变量是型态为MSG的结构，型态MSG在WINUSER.H中定义如下：  
typedef struct tagMSG         
{         
    HWND   hwnd ;          
    UINT   message ;          
    WPARAM wParam ;          
    LPARAM lParam ;          
    DWORD  time ;          
    POINT  pt ;         
}         
MSG, \* PMSG ;  
        
POINT数据型态也是一个结构，它在WINDEF.H中定义如下：  
typedef struct tagPOINT         
{         
    LONG  x ;         
    LONG  y ;         
}         
POINT, \* PPOINT;  
TranslateMessage(&msg); 将msg结构传给Windows，进行一些键盘转换。（关于这一点，我们将在第六章中深入讨论。）  
DispatchMessage(&msg);又将msg结构回传给Windows。然后，Windows将该消息发送给适当的窗口消息处理程序，让它进行处理。这也就是说，Windows将呼叫窗口消息处理程序。在HELLOWIN中，这个窗口消息处理程序就是WndProc函数。处理完消息之后，WndProc传回到Windows。此时，Windows还停留在DispatchMessage呼叫中。在结束DispatchMessage呼叫的处理之后，Windows回到HELLOWIN程序中，并且接着从下一个GetMessage呼叫开始消息循环。

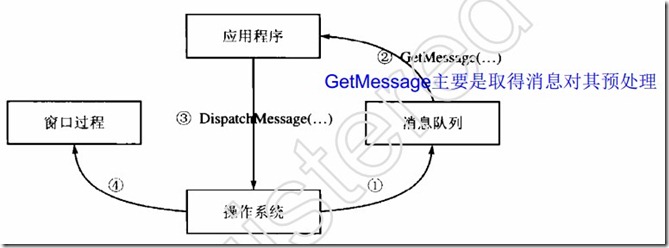
附：关于窗口过程和消息循环的一些注解。转自孙鑫老师《Windows程序内部运行机制》

1.关于WNDPROC函数指针原型：typedef LRESULT (CALLBACK \* WNDPROC)(HWND, UINT, WPARAM, LPARAM);

   其中CALLBACK函数约定就是\_\_stdcall约定.

2.TranslateMessage函数用于将虚拟键消息转为字符消息。字符消息被投递到调用线程的消息队列中，当下一次调用GetMessage函数时取出。当我们敲击键盘上的某个字符键时，系统将产生WM\_KEYDOWN和WM\_KEYUP消息。这两个消息的附加参数（wParam和lParam）包含的是虚拟键代码和扫描码等消息，而我们在程序中往往需要得到某个字符的ASSII码，TranslateMessage这个函数可以将WM\_KEYDOWN和WM\_KEYUP消息的组合转换为一条WM\_CHAR消息（该消息的wParam附加参数包含了字符的ASCII码），并将转换后的新消息投递到调用线程的消息队列中。注意，TranslateMessage函数并不会修改原有的消息，它只是产生新的消息并投递到消息队列中。

DispatchMessage函数分派一个消息到窗口过程，由窗口过程函数对消息进行处理。DispatchMessage实际上是将消息回传给操作系统，由操作系统调用窗口过程函数对消息进行处理（响应）。

Windows应用程序的消息处理机制如下图所示：

1）操作系统收到应用程序的窗口消息，将消息投递到该应用程序的消息队列中。

（2）应用程序在消息循环中调用GetMessage函数从消息队列中取出一条一条的消息。取出消息后，应用程序可以对消息进行一些预处理，例如，放弃对某些消息的响应，或者调用TranslateMessage产生新的消息。

（3）应用程序调用DispatchMessage，将消息回传给操作系统。消息是由MSG结构体对象进行表示的，其中就包含了接收消息的窗口的句柄。因此，DispatchMessage函数总能进行正确的传递。

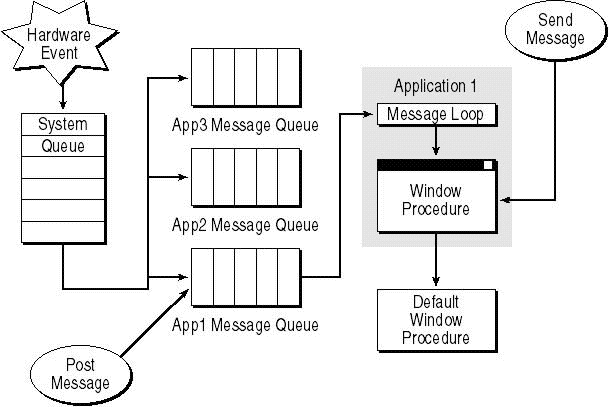
（4）系统利用WNDCLASS结构体的lpfnWndProc成员保存的窗口过程函数的指针调用窗口过程，对消息进行处理（即“系统给应用程序发送了消息”）。

以上就是Windows应用程序的消息处理过程。

三 队列化消息与非队列化消息  
      
消息能够被分为「队列化的」和「非队列化的」。队列化的消息是由Windows放入程序消息队列中的。在程序的消息循环中，重新传回并分配给窗口消息处理程序。非队列化的消息在Windows呼叫窗口时直接送给窗口消息处理程序。也就是说，队列化的消息被「发送」给消息队列，而非队列化的消息则「发送」给窗口消息处理程序。任何情况下，窗口消息处理程序都将获得窗口所有的消息--包括队列化的和非队列化的。窗口消息处理程序是窗口的「消息中心」。

队列化消息基本上是使用者输入的结果，以击键（如WM\_KEYDOWN和WM\_KEYUP消息）、击键产生的字符（WM\_CHAR）、鼠标移动（WM\_MOUSEMOVE）和鼠标按钮（WM\_LBUTTONDOWN）的形式给出。队列化消息还包含时钟消息（WM\_TIMER）、更新消息（WM\_PAINT）和退出消息（WM\_QUIT）。

非队列化消息则是其它消息。在许多情况下，非队列化消息来自呼叫特定的Windows函数。例如，当WinMain呼叫CreateWindow时，Windows将建立窗口并在处理中给窗口消息处理程序发送一个WM\_CREATE消息。当WinMain呼叫ShowWindow时，Windows将给窗口消息处理程序发送WM\_SIZE和WM\_SHOWWINDOW消息。当WinMain呼叫UpdateWindow时，Windows将给窗口消息处理程序发送WM\_PAINT消息。键盘或鼠标输入时发出的队列化消息信号，也能在非队列化消息中出现。例如，用键盘或鼠标选择了一个菜单项时，键盘或鼠标消息就是队列化的，而说明菜单项已选中的WM\_COMMAND消息则可能就是非队列化的。

四 SendMessage()与PostMessage()之间的区别是什么？  
它们两者是用于向应用程序发送消息的。PostMessagex()将消息直接加入到应用程序的消息队列中，不等程序返回就退出；而SendMessage()则刚好相反，应用程序处理完此消息后，它才返回。我想下图能够比较好的体现这两个函数的关系：

五 函数peekmessage和getmessage的区别？

两个函数主要有以下两个区别:  
1.GetMessage将等到有合适的消息时才返回,而PeekMessage只是撇一下消息队列。  
2.GetMessage会将消息从队列中删除,而PeekMessage可以设置最后一个参数wRemoveMsg来决定是否将消息保留在队列中。