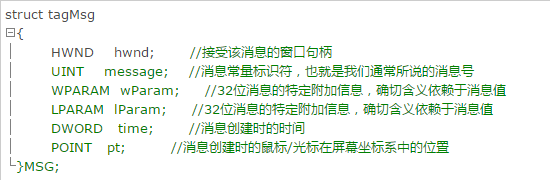
1. 消息是指什么?

消息系统对于一个win32程序来说是十分重要的,它是一个程序运行的动力的源泉。一个消息，是系统定义的一个32位的值，它唯一的定义了一个事件，向windows发出一个通知，告诉应用程序某个事情发生了。例如，单机鼠标、改变窗口尺寸、按下键盘上的一个键都会使windows发送一个消息给应用程序。

消息本身是作为一个记录传递给应用程序的，这个记录中包含了消息的类型以及其他信息。例如，对于单击鼠标所产生的消息来说，这个记录中包含了单击鼠标时的坐标。这个记录类型叫做“MSG”，MSG包含有来自windows应用消息队列的消息信息，它在windows中的声明如下：



消息可以用系统或者应用程序产生。系统在发生输入事件时产生消息。举例：当用户敲键，移动鼠标或单击控件。系统也产生消息以响应由应用程序带来的变化，比如应用程序改变系统字体改变窗口大小。应用程序可以产生消息使窗体执行任务，或者与其他应用程序中的窗口通讯。

1. 消息有哪几种？

Windows中的消息虽然很多，但种类并不复杂，大体上有3种：窗口消息、命令消息和控件通知消息。

a. 窗口消息：大概是系统中最为常见的消息，它是由操作系统和控制其他窗口的窗口所使用的消息。例如CreateWindow、DestroyWIndows和MoveWindow等都会激发窗口消息，还有我们再上面谈到的单击鼠标所产生的消息也是一种窗口消息。

b. 命令消息：这是一种特殊的窗口消息，他用来处理从一个窗口发送到另一个窗口的用户请求，例如按下一个按钮，他就会向主窗口发送一个命令消息。

c. 控件通知消息: 是指这样一种消息,一个窗口内的子控件发生了一些事情,需要通知父窗口.通知消息只适用于标准的窗口控件如按钮、列表框、组合框、编辑框、以及windows公共控件如树状视图、列表视图等。

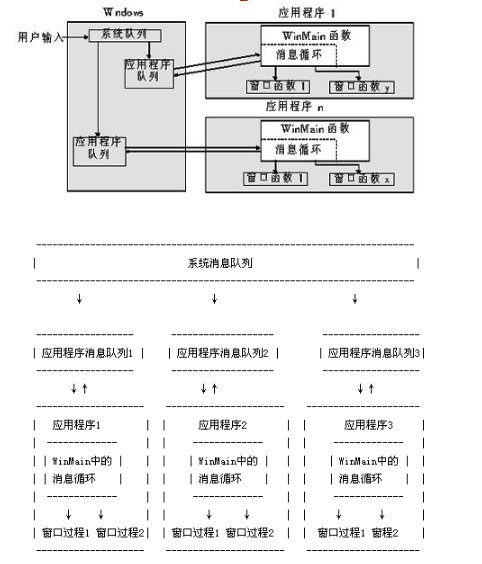
窗口消息和控件通知消息主要由窗口类即直接或间接由CWND类和其派生类处理。相对于窗口消息和控件通知消息而言，命令消息的处理对象范围就广的多，它不仅可以由窗口类处理，还可以由文档类，文档模板类以及应用类去处理。

1. 队列消息和非队列消息：

从消息的发起途径来看，消息可以分为2种：队列消息和非队列消息。消息队列又可以分成系统消息队列和线程消息队列。系统消息队列由windows维护，线程消息队列则由每个GUI线程自己进行维护，为避免给non-GUI线程创建消息队列，所有线程产生时并没有消息队列，仅当线程第一次调用GDI函数时系统才给线程创建消息队列。队列消息送到系统消息队列，然后到线程消息队列；非队列消息直接送给目的窗口过程。

对于队列消息，最常见的是鼠标和键盘触发的消息，例如WM\_MOUSERMOVE、WM\_CHAR等消息，还有一些其他的消息，例如WM\_PAINT、WM\_TIMER和WM\_QUIT。当鼠标、键盘事件被触发后，相应的鼠标或键盘驱动程序就会把这些事件转换成相应的消息，然后输送到系统消息队列，由windows系统去进行处理。Windows系统则在适当的时机，从系统消息队列中取出一个消息，根据前面我们所说的MSG消息结构确定是要被送往哪个窗口，然后把取出的消息送往创建窗口的线程的相应队列，下面的事情就该由线程消息队列操心了，windows开始忙于自己的事情去了。线程看到自己的消息队列中有消息，就从队列中取出来，通过操作系统发送到合适的窗口过程去处理。

一般来讲，系统总是将消息Post在消息队列的末尾。这样保证窗口以先进先出的顺序接收消息。然而，WM\_PAINT是一个例外，同一个窗口的多个VM\_PAINT被合并成一个WM\_PAINT消息，合并所有的无效区域到一个一个无效区域。合并WM\_PAINT的目的是为了减少刷新窗口的次数。



非消息队列将会绕过系统队列和消息队列，直接将消息发送到窗口过程。例如，当用户激活一个窗口系统发送WM\_ACTIVATE、WM\_SETFOCUS、WM\_SETCURSOR。这些消息通知窗口它将被激活了。非队列消息也可以由应用程序调用系统函数产生。例如，当程序调用SetWindowPos系统发送WM\_WINDOWPOSCHANGED消息。一些函数也发送非队列消息，例如下面我们要谈的函数。

1. 消息的发送：

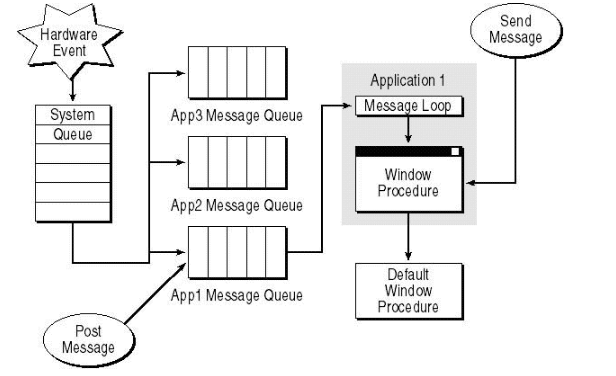
把一个消息发送到窗口的方式有3种： 发送、寄送和广播。

发送消息的函数有SendMessage、SendMessageCallback、SendNotifyMessage、SendMessageTimeout;寄送消息的函数主要有PostMessage、PostThreadMessage、PostQuitMessage；广播消息的函数主要有BroadcastSystemMessage、BroadcastSystemMessageEx。

SendMessage的原型如下：LRESULT SendMessage(HWND hWnd,UINT Msg,WPARAM wParam,LPARAM lParam)，这个函数主要是向一个或多个窗口发送一条消息，一直等到消息被处理才返回。不过需要注意的是：如果接收消息的窗口是同一个应用程序的一部分（即同一个线程中），那么这个窗口的窗口函数就被作为一个子程序马上被调用；如果接收消息的窗口是被另外的线程所创建的，那么窗口系统就切换到相应的线程并且调用相应的窗口函数，这条消息不会被放进目标应用程序的队列中。函数的返回值是由接收消息的窗口的窗口函数返回，返回的值取决于被发送的消息。

PostMessage的原型如下：BOOL PostMessage(HWND hWnd,UINT Msg,WPARAM wParam,LPARAM lParam)，该函数把一条消息放置到创建hWnd窗口的线程的消息队列中，该函数不等消息被处理就马上将控制返回。需要注意的是：如果hWnd参数为HWND\_BROADCAST，那么，消息将被寄送给系统中的所有的重叠窗口和弹出窗口，但是子窗口不会接收到该消息；如果hWnd参数为NULL，则该函数类似于将dwThreadID参数设置成当前线程的标志来调用PostThreadMessage函数。

以上这两个代表性的函数，我们可以看出消息的发送方式和寄送方式的区别在于：被发送的消息是会被立即处理，函数是否立即返回。SendMessage消息会被立即处理，处理完后函数才会返回；PostMessage消息不会被立即处理，他会被先放进一个先进先出的队列中，一直等待应用程序空闲的时候才会被处理，不过函数放置消息后立即返回。



实际上，发送消息到一个窗口处理过程和直接调用窗口处理过程之间并没有太大的区别，他们直接的唯一区别在于：你可以要求操作系统截获所有被发送的消息，但是不能够截获对窗口处理过程的直接调用。

寄送方式发送消息的应用场景：通常是与用户输入事件相对应的，因为这些事件不是十分紧迫，可以进行缓慢的缓冲处理，例如鼠标、键盘消息会被寄送，而按钮消息则会被发送。

广播消息用的比较少，BroadcastSystemMessage函数的原型如下：long BroadcastSystemMessage(DWORD dwFlags,LPDWORD lpdwRecipients,UINT uiMessage,WPARAM wParam,LPARAM lParam);该函数可以向指定的接受者发送一条消息，这些接受者可以是应用程可安装的驱动程序、网络驱动程序、系统级别的设备驱动程序和他们的任意组合。需要注意的是：如果dwFlags参数是BSF\_QUERY并且至少一个接收者返回了BROADCASET\_QUERY\_DENY，则返回至为0，如果没有指定BF\_QUERY，则函数将消息发送给所有接收者，并且忽略其返回值。

1. 消息的接收：

消息的接收注意有3和函数：GetMessage、PeekMessage、WaitMessage。

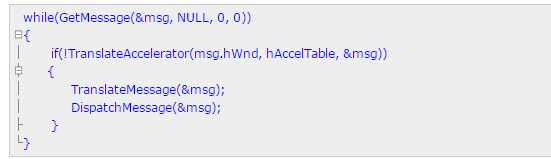
GetMessage原型如下：BOOL GetMessage(LPMSG lpMsg,HWND hWnd,UINT wMsgFilterMin,UINT wMsgFilterMax);该函数用来获取与hWnd参数所指定的窗口相关的且wMsgFilterMin和wMsgFilterMax参数所给出的消息值范围内的消息。需要注意的是：如果hWnd为NULL，则GetMessage获取属于调用该函数应用的程序的任意窗口的消息，如果wMsgFilterMin和wMsgFilterMax都是0，则GetMessage就返回所有可得到的消息。函数获取之后将删除消息队列中的除WM\_PAINT消息之外的其他消息，至于WM\_PAINT则只有在其处理之后才被删除。

PeekMessage原型如下：BOOL PeekMessage(LPMSG lpMsg,HWND hWnd,UINT wMsgFilterMin,UINT wMsgFilterMax,UINT wRemoveMsg）；该函数用于查看应用程序的消息队列，如果其中有消息就将其放入lpMsg所指的结构中。不过，与GetMessage不同的是：PeekMessage函数不会等到有消息放入队列时才返回。同样，如果hWnd为NULL，则PeekMessage获取属于调用该函数的应用程序的任一窗口的消息，如果hWnd=-1，那么函数只返回hWnd参数为NULL的PostMessage函数发送去的消息。如果wMsgFilterMin和wMsgFilterMax都是0，则PeekMessage就返回所有可得到的消息。函数获取之后将视最后一个参数来决定是否删除消息队列中的除WM\_PAINT消息之外的其他消息，至于WM\_PAINT则只有在其处理之后才被删除。

WaitMessage原型如下：BOOL WaitMessage();当一个应用程序无事可做时，该函数就将控制权交给另外的应用程序，同时将该应用程序挂起，直到一个新的消息被放入应用程序的队列中才返回。

1. 消息的处理：

我们来看一下VC中的消息泵：



首先，GetMessage从进程的主线程的消息队列中获取一个消息并将它复制到MSG结构，如果队列中没有消息，则GetMessage函数将等待一个消息的到来以后才返回。如果你将一个窗口句柄作为第二个参数传入GetMessage，那么只有指定窗口的的消息可以从队列中获得。GetMessage也可以从消息队列中过滤消息只接受消息队列中落在范围内的消息。这时候就要利用GetMessage／PeekMessage指定一个消息过滤器。这个过滤器是一个消息标识符的范围或者是一个窗体句柄，或者两者同时指定。当应用程序要查找一个后入消息队列的消息是很有用。WM\_KEYFIRST 和 WM\_KEYLAST 常量用于接受所有的键盘消息。 WM\_MOUSEFIRST 和 WM\_MOUSELAST 常量用于接受所有的鼠标消息。   
　 然后TranslateAccelerator判断该消息是不是一个按键消息并且是一个加速键消息，如果是，则该函数将把几个按键消息转换成一个加速键消息传递给窗口的回调函数。处理了加速键之后，函数TranslateMessage将把两个按键消息WM\_KEYDOWN和WM\_KEYUP转换成一个 WM\_CHAR，不过需要注意的是，消息WM\_KEYDOWN,WM\_KEYUP仍然将传递给窗口的回调函数。       
　 处理完之后，DispatchMessage函数将把此消息发送给该消息指定的窗口中已设定的回调函数。如果消息是WM\_QUIT，则 GetMessage返回0，从而退出循环体。应用程序可以使用PostQuitMessage来结束自己的消息循环。通常在主窗口的 WM\_DESTROY消息中调用。

1. 窗口过程

窗口过程是一个用于处理所有发送到这个窗口的消息的函数。任何一个窗口类都有一个窗口过程，同一个类的窗口使用同样的窗口过程来响应消息。系统发送消息给窗口过程将消息数据作为参数传递给他，消息到来之后，按照消息类型排序进行处理，其中的参数则用来区分不同的消息，窗口过程使用参数产生合适的行为（类似于消息分流器）。

一个窗口过程不经常忽略消息，如果他不处理，它会将消息传回到执行默认的处理。窗口过程通过调用DefWindowProc函数来做这个处理。窗口过程必须return一个值作为它的消息处理结果。大多数窗口只处理小部分消息。其他的将通过DefWindowProc传递给系统做默认处理。窗口过程被所有属于同一个类的窗口共享，能为不同窗口处理消息。

参考文章: https://www.cnblogs.com/zhoug2020/p/6239018.html