　Windows的应用程序是事件(消息)驱动的。它们不会显式地调用函数（如C运行时库调用）来获取输入，而是等待windows向它们传递输入。 windows系统把应用程序的输入事件传递给各个窗口，每个窗口有一个函数，称为窗口消息处理函数。窗口消息处理函数处理各种用户输入，处理完成后再将控制权交还给系统。窗口消息处理函数一般是在注册一个窗口的时候指定的。

0x01  **Windows消息**

　　windows通过消息的形式向窗口传递用户输入。消息可以由系统和应用程序生成。该系统会为每个输入事件产生相应的消息，例如，用户点击鼠标，移动鼠标或滚动条，或是应用程序改变了系统的某些属性，比如说系统更改了字体资源，改变了某个窗口的大小。 不仅如此，应用程序可以生成消息，通告发送消息指定它的窗体去执行某些任务或者是与其他的应用程序交互。

　　windows系统将消息发送到一个窗口消息处理函数时传递四个参数：窗口句柄，消息标识符，两个DWORD值(消息参数)。窗口句柄标识了该消息的目的窗口。windows使用它来确定是哪个窗口的的窗口消息处理函数收到该消息。一个消息标识符是一个有名字的常量，用来表明消息的意义。当一个窗口处理函数收到一条消息，它根据判断消息标识符来决定如何处理该消息，例如，消息标识符WM\_PAINT消息告诉窗口程序窗口的客户区已发生变化，必须重绘。 消息参数(DWORD值)指定传递的数据或是数据的地址。消息参数可以是一个整型值，一个指针值。也可以为NULL。一个窗口过程必须根据消息标识符来确定如何解释消息参数。

0x02  **windows 消息类型**

　　消息的两种类型：

　　(1) 系统定义的消息

　　(2) 应用程序定义的消息

**系统定义的消息**

　　操作系统向应用程序发送消息来和应用程序通讯。操作系统通过消息控制应用程序的运行，向应用程序传递用户输入以及一些其他有用的信息。

　　应用程序也可以发送系统定义的消息，应用程序通过这些消息去控制使用注册窗口类创建的控件的窗口的运行。

　　每个系统定义的消息都有一个唯一的消息标识符和相应的符号常量（在windows SDK的头文件里定义）。符号常量通常会表明系统定义的消息所属的类别。不同的前缀表明不同的类别。一下是常见的分类：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | Prefix Message category      WM        General window(一般的窗口)      ABM        Application desktop toolbar (应用程序桌面工具条)      BM        Button control (按钮控件)      CB        Combo box control (组合框控件)      CBEM Extended combo box control(扩展的组合框控件)      CDM        Common dialog box (普通的对话框)      DBT        Device (设备)      DL        Drag list box (下拉列表)      DM        Default push button control (默认按钮控件)      DTM        Date and time picker control(日期和时间选择控件)      EM        Edit control (编辑控件)      HDM        Header control (表头控件)      HKM        Hot key control (热键控件)      IPM        IP address control (IP地址控件)      LB        List box control  (列表框控件)      LVM        List view control (列表视图控件)      MCM        Month calendar control (数学日历控件)      PBM        Progress bar (进度条控件)      PGM        Pager control ()      PSM        Property sheet (属性页)      RB        Rebar control (分隔条控件)      SB        Status bar window (状态条控件)      SBM        Scroll bar control (滚动条控件)      STM        Static control (静态控件)      TB        Toolbar (工具条)      TBM        Trackbar (跟踪栏)      TCM        Tab control (选项卡控件)      TTM        Tooltip control ()      TVM        Tree-view control ()      UDM        Up-down control () |

　　应用程序可以通过创建自定义的消息，用来和自己的窗口和其他进程通讯。如果应用程序创建了自己的消息，窗口处理函数可以解析这些信息，并作出相应的处理。

　　消息标识符值的取值范围：

　　该系统保留了一个消息范围，从0x0000到0x03FF(0x03FF等于WM\_USER -1)范围. 这个范围内的值为系统定义的消息。应用程序不能使用这些值作为自己的自定义消息。从0x0400（数值WM\_USER)到0x7FFF的值是为应用程序保留的。应用程序可以使用这个范围内的值来定义自己的消息。如果操作系用的版本(windows version)主版本为4.0版，你还可以使用0x8000(WM\_APP)到0xBFF之间的值来定义自己的消息。除此之外，应用程序还可以调用RegisterWindowMessage函数注册一个消息时，操作系统会返回一个介于0xC000和0xFFFF之间的一个消息标识符。并且保证这个返回值是系统唯一的。因此，可以避免和其他应用程序使用的消息相冲突。

0x03   **消息派发**

　　windows使用两种方法将消派发到一个窗口消息处理函数：一是将消息放到消息队列(先进先出队列)，二是不放到消息队列，直接发送到窗口消息处理函数，让窗口处理函数来处理消息。派发到消息队列的消息被称为排队消息(Queued messages)。它们主要是用户输入事件，比如说鼠标或键盘消息盘，有WM\_MOUSEMOVE消息，WM\_LBUTTONDOWN，WM\_KEYDOWN，和WM\_CHAR消息。还有一些其他的，包括WM\_TIMER，WM\_PAINT，以及WM\_QUIT。大多数其他的消息息，这是直接发送到窗口过程，被称为非队列消息(non queued messages)。

　　(1) 队列(Queued)消息

　　windows可同时显示任意数量的窗口。此时，系统使用消息队列来将键盘和鼠标事件正确的派发到正确的窗口。

　　windows维护着一个系统消息队列，以及分别为每个GUI线程维护一个各自的线程消息队列。为了避免非GUI线程的创建线程消息队列的开销，所有线程创建初始化时，均不创建消息队列。只有当线程第一次调用GDI函数时，系统才会为线程创建消息队列。所以那些非GUI线程是没有消息队列的。每当用户移动鼠标，点击按钮或键盘时，鼠标或键盘的设备驱动程序会将输入转换成消息，并将消息放在系统消息队列里。然后windows会检查自己的消息队列，如果消息队列不为空，则每次取出并删除一个消息，然后确定消息的目标窗口，然后把消息放到创建这个窗口的线程的线程消息队列里。线程的消息队列接收由线程创建的窗口的所有的鼠标和键盘消息。然后线程会从队列中删除信息，并告诉系统把它们派发到对应的窗口消息处理函数。除了WM\_PAINT, WM\_TIMER和WM\_QUIT消息以外，系统总是派发放在在消息队列的末尾的消息。这将保证让一个窗口以first-in, first-out的顺序接收消息。WM\_PAINT，WM\_TIMER，和WM\_QUIT消息，会一直被保存在队列中，只有在队列中没有其他消息时才会被派发到窗口消息处理函数。此外，同一个窗口的多个WM\_PAINT消息被合并成一个WM\_PAINT消息，客户区的所有无效部分也会被合并。这样是为了减少窗口重绘客户区的次数。windows向线程消息队列传递消息时，首先会填充一个MSG结构，然后将这个MSG结构复制到消息队列。MSG中的信息包括：目标窗口，消息标识符，两个消息参数，消息派发时的时间，鼠标光标位置。一个线程可以使用PostMessage或PostThreadMessage功能向自己的消息队列或者是其他线程的消息队列发送消息。应用程序可以使用GetMessage函数从自己的消息队列中删除消息。查看而不删除消息，用的是PeekMessage函数。PeekMessage函数会返回一个带有消息信息的MSG结构。从消息队列中删除消息后，应用程序可以使用DispatchMessage函数指示系统将消息发送到一个窗口消息处理函数。 DispatchMessage的参数是是前一次调用GetMessage或PeekMessage获得的MSG结构的指针。 DispatchMessage会传递窗口句柄，消息标识符，这两个消息参数这些信息给窗口消息处理函数，它不会传递消息派发时间以及鼠标光标位置。应用程序可以在处理消息时调用的GetMessageTime和GetMessagePos来获得这些信息。线程可以使用WaitMessage函数，交出自己的控制权，当它的消息队列中没有消息时，调用WaitMessage函数会挂起线程，直到自己的消息队列里有消息时才返回。

　　可以调用SetMessageExtraInfo函数来关联一个值给当前线程的消息队列。然后调用GetMessageExtraInfo函数来获取由GetMessage或PeekMessage函数得到的最后一条消息相关联的值。你可以去msdn上看更多的关于这几个函数的信息。

　　(2) 非队列(Nonqueued)消息

　　Nonqueued消息被立即送往目的地的窗口消息处理函数，绕过了系统的消息队列和线程消息队列。系统通常会发送nonqueued消息，来通知那些会影响窗口的事件。例如，当用户激活一个新的应用程序窗口时，系统会发送一些列消息到窗口，包括WM\_ACTIVATE，WM\_SETFOCUS，WM\_SETCURSOR。这些消息通知窗口被激活，键盘输入被定向到窗口，并且鼠标光标也移到窗口的边界内。Nonqueued消息也有可能来源于应用程序调用系统函数。例如，系统调用SetWindowPos函数移动一个窗口后会发送WM\_WINDOWPOSCHANGED消息。 一些函数也发送nonqueued消息， 有BroadcastSystemMessage，BroadcastSystemMessageEx，SendMessage，SendMessageTimeout，和SendNotifyMessage。

0x04   **消息处理**

　　应用程序必须删除并处理发送到它的线程消息队列的消息。单线程应用程序通常在它的WinMain函数的消息循环，删除和分发消息到适当的窗口进行处理。多线程应用程序可以在每一个线程创建一个窗口的消息循环。消息循环如何工作，并讲述窗口消息处理函数的作用：

　　(1)消息循环

　　(2)窗口处理函数

**消息循环**

　　一个简单的消息循环包含调用以下三个函数：GetMessage，TranslateMessage，和DispatchMessage。请注意，如果有一个错误，GetMessage返回-1 -因此，需要测试它的返回值，来判断为-1的情况.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | MSG msg;  BOOL bRet;  while( (bRet = GetMessage( &msg, NULL, 0, 0 )) != 0)  {      if (bRet == -1)      {          // handle the error and possibly exit      }      else      {          TranslateMessage(&msg);          DispatchMessage(&msg);      }  } |

　　GetMessage函数从队列中获取消息，并将消息内容复制到一个MSG结构。它返回一个非零值，除非遇到WM\_QUIT消息，此种返回FALSE并结束消息循环。在一个单线程应用程序，结束消息循环往往是在关闭应用程序的第一步。应用程序可以调用PostQuitMessage函数来响应WM\_DESTROY，结束消息循环。

如果指定一个窗口句柄作为GetMessage的第二个参数，那么GetMessage只获取在消息队列里和这个窗口有关的消息。 GetMessage也可以在队列中筛选消息，只获取指定范围内的消息。如需有关消息过滤的详细信息，请参考消息过滤。

　　线程的消息循环必须包括TranslateMessage，如果线程需要接受键盘字符的输入。每次用户按下一个键，该系统产生相应的虚拟键消息（WM\_KEYDOWN和WM\_KEYUP）。虚拟键消息包含一个虚拟键码，标识的是被按下的键，而不是它相关的字符值。要获得此值，消息循环必须包含TranslateMessage，用来将拟键消息翻译成字符消息（WM\_CHAR）的并放到应用程序的消息队列里。经过若干次循环后，WM\_CHAR消息会被并派发到一个窗口。DispatchMessage函数将消息发送到到与MSG结构中的窗口句柄关联的窗口。如果窗口句柄是HWND\_TOPMOST，DispatchMessage则将消息发送到操作系统所有的顶层窗口。如果窗口句柄是NULL，DispatchMessage不做任何事。一个应用程序的主线程初始化后，系统就启动应用程序的消息循环，并创造至少一个窗口。一旦启动，消息循环持续从该线程的消息队列中删除消息，并派发他们到相应的窗口。GetMessage函数从消息列表中获取到WM\_QUIT消息时，消息循环结束。一个消息队列只需要一个消息循环，即使一个应用程序包含有多个窗口。 DispatchMessage总是调度消息到正确的窗口，这是因为每个队列中的消息是MSG结构，它包含着消息所属的窗口的句柄。可以以多种方式来修改消息循环。例如，您可以从队列中删除消息，但是不派发他们。当发送有些不带有目的地窗口的消息时这非常有用。您也可以使用GetMessage只获取指定的消息，这是有用的，如果你必须你暂时绕过正常的消息队列FIFO顺序。

　　应用程序使用快捷键时，必须能够将键盘消息转换为命令消息。因此，应用程序的消息循环必须包括TranslateAccelerator函数调用。关于快捷键的更多信息，请参见键盘加速器。

　　如果一个线程使用一个无模式对话框，那么消息循环必须包括IsDialogMessage函数，以便该对话框可以接收键盘输入。

　　(3)窗口消息处理函数

　　窗口消息函数接收和处理的所有发送到窗口的消息。每个窗口类有一个窗口消息处理函数，用该类创建的每个窗口使用同一窗口消息处理函数。

　　该系统将消息发送到一个窗口的程序，并传递消息的相关信息到窗口消息处理函数，窗口消息处理函数检查消息标识符，根据传过来的参数识别并处理不同的消息，一个窗口过程通常不会忽略一个消息。如果消息没有被处理，必须被发送给系统默认的窗口消息处理函数，这是否通过调用DefWindowProc函数，来执行一个默认的处理，并返回一个处理的结果。窗口程序必须然后返回该值作为自己的消息处理的结果。大多数窗口消息处理函数只处理一少部分消息，并将其他的返回给系统默认的窗口消息处理函数。

　　因为窗口消息处理函数被所有属于同一个窗口类的窗口共享，它可以处理几个不同的窗口的消息。要确定具体的窗口消息，窗口消息处理函数可以检查消息结构里的窗口句柄。

分类: [Windows 核心编程](https://www.cnblogs.com/lsh123/category/1053525.html)