

朝闻道

朝闻道，夕可死矣！为了成为IT高手，为了挽回我失去的青春，也为了我亲爱的家人，下决心刻苦学习编程知识，虽九死而不悔！金头盔飞行员蒋佳翼：知道了不行，熟悉也不够，要真正进入潜意识，成为条件反射才行。

CnBlogs Home New Post Contact Admin Rss Posts - 3258 Articles - 10 Comments - 43

Search

Post Categories

Architecture-CPU(18)
ASM-Exe文件(2)
ASM-经典资料(7)
ASM-学习(12)
ASM-指针(13)
BCB(8)
C++ Library(16)
C++ LLVM心得(4)
C++ MinGW
C++ OO研究(30)
C++ RTL(13)
C++ STL(2)
C++ 编译(5)
C++ 构造与析构(15)
C++ 关键字(5)
C++ 函数调用(20)
C++ 内存(67)
C++ 学习(14)
C++ 哲学(4)
CSharp(4)
Delphi-ASM(16)
Delphi-COM(23)
Delphi-Compiler(9)
Delphi-DataType(24)
Delphi-Editor(23)
Delphi-Exception(13)
Delphi-Exe(34)
Delphi-FireMonkey配置(31)
Delphi-FireMonkey源码(28)
Delphi-GDI(21)
Delphi-MIS设计(9)
Delphi-OO研究(31)
Delphi-Process(10)
Delphi-RTL(14)
Delphi-RTTI(23)
Delphi-String(23)
Delphi-Thread(29)
Delphi-Unidac(5)
Delphi-Unigui(3)
Delphi-VCL窗体(59)
Delphi-VCL控件创建与显示(52)
Delphi-VCL源码研究(39)
Delphi-WebBrowser(37)
Delphi-WindowState(11)
Delphi-x64(14)
Delphi-XML-JSON(14)
Delphi-高手突破(5)
Delphi-经典资料(27)
Delphi-经典资料-Install(13)
Delphi-控件开发(27)
Delphi-控件使用(10)
Delphi-内存(33)
Delphi-三层(19)
Delphi-时间(24)
Delphi-时间器(17)
Delphi-事件(20)
Delphi-数据库(6)
Delphi-图形图像(27)

windows消息机制（MFC）

windows消息机制（MFC）

消息分类与消息队列

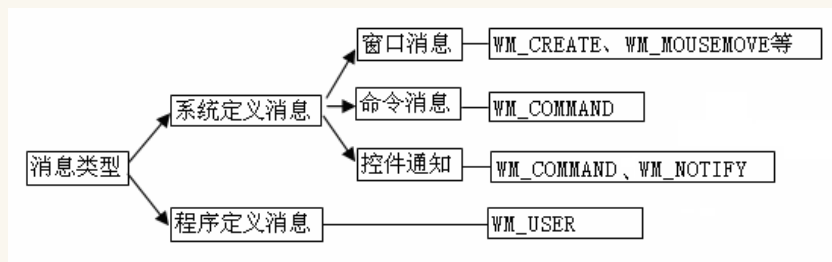
Windows中，消息使用统一的结构体（MSG）来存放信息，其中message表明消息的具体的类型，

而wParam，lParam是其最灵活的两个变量，为不同的消息类型时，存放数据的含义也不一样。

time表示产生消息的时间，pt表示产生消息时鼠标的位置。

```
typedef struct tagMSG
{
    HWND hwnd;
    UINT message;
    WPARAM wParam;
    LPARAM lParam;
    DWORD time;
    POINT pt;
} MSG;
```

按照类型，Windows将消息分为：



(0) 消息ID范围

系统定义消息ID范围：[0x0000, 0x03ff]

用户自定义的消息ID范围：

WM_USER: 0x0400-0x7FFF (例：WM_USER+10)

WM_APP(winver> 4.0): 0x8000-0xBFFF (例：WM_APP+4)

RegisterWindowMessage: 0xC000-0xFFFF 【用来和其他应用程序通信，为了ID的唯一性，使用::RegisterWindowMessage来得到该范围的消息ID】

(1) 窗口消息：即与窗口的内部运作有关的消息，如创建窗口，绘制窗口，销毁窗口等。

可以是一般的窗口，也可以是MainFrame,Dialog,控件等。

如：WM_CREATE, WM_PAINT, WM_MOUSEMOVE, WM_CTLCOLOR, WM_HSCROLL等

(2) 当用户从菜单选中一个命令项目、按下一个快捷键或者点击工具栏上的一个按钮，都将发送WM_COMMAND命令消息。

LOWORD(wParam)表示菜单项，工具栏按钮或控件的ID；如果是控件，HIWORD(wParam)表示控件消息类型。

```
#define LOWORD(l) ((WORD)(l))
```

```
#define HIWORD(l) (((DWORD)(l) >> 16) & 0xFFFF)
```

(3) 随着控件的种类越来越多，越来越复杂（如列表控件、树控件等），仅仅将wParam，lParam将视为一个32位无符号整数，已经装不下太多信息了。

为了给父窗口发送更多的信息，微软定义了一个新的WM_NOTIFY消息来扩展WM_COMMAND消息。

WM_NOTIFY消息仍然使用MSG消息结构，只是此时wParam为控件ID，lParam为一个NMHDR指针，

不同的控件可以按照规则对NMHDR进行扩充，因此WM_NOTIFY消息传送的信息量可以相当的大。

注：Window 9x 版及以后的新控件通告消息不再通过WM_COMMAND 传送，而是通过WM_NOTIFY 传

Delphi-网络(28)
Delphi-文件(31)
Delphi-系统编程(25)
Delphi-消息WM_PAINT(30)
Delphi-消息大全(32)
Delphi-消息研究(54)
Delphi-消息应用(34)
Delphi-语法(3)
Delphi-哲学(19)
Delphi-指针(9)
Golang-学习(2)
Golang-哲学(12)
HTTP-TCP等各种协议(21)
IC-仿真(5)
Java-OO研究(6)
Java-学习(5)
Linux-服务器(61)
Linux-工具(14)
Mac(10)
Mobile-哲学(7)
MySQL(26)
OpenGL(3)
OpenSSL(12)
Python-wxPython(8)
Python-语法(3)
Qt-COM(13)
Qt-Layout(5)
Qt-Lib-DLL(24)
Qt-Model View(42)
Qt-Process(24)
Qt-Python(3)
Qt-QML(9)
Qt-QSS(21)
Qt-String(48)
Qt-Thread(32)
Qt-安装与编译(123)
Qt-经典资料(56)
Qt-控件(114)
Qt-容器(11)
Qt-事件(53)
Qt-数据库(20)
Qt-图形图像(4)
Qt-网络编程(43)
Qt-文件(20)
Qt-信号槽(28)
Qt-学习(25)
Qt-源码(14)
Qt-哲学(12)
SaaS(55)
Saas-Job(33)
SaaS-Soft(55)
SaaS-Tech(15)
SaaS-起居注(38)
SaaS-云存储, 云服务(47)
SG3-编程经验(29)
SG3-行业经验(25)
VC++ DirectUI(9)
VC++ Editor(2)
VC++ MFC(17)
VC++ String编码理论(22)
VC++ String编码应用(20)
VC++ 经典资料(28)
VC++ 图形图像(14)
VC++ 哲学(4)
VirtualBox(25)
Web-HTML(3)
Web-JavaScript(15)
Web-JQuery(2)
Web-PHP(10)
Web-经典(7)
Web-系统(6)
Windows-API(8)
Windows-DLL(3)
Windows-SDK(9)
Windows-UAC与数字签名(49)
Windows-窗口(5)

送，
但是老控件的通告消息， 比如CBN_SELCHANGE 还是通过WM_COMMAND 消息发送。

(4) windos也允许程序员定义自己的消息，使用SendMessage或PostMessage来发送消息。

windows消息还可以分为：

(1) 队列消息(Queued Messages)

消息会先保存在消息队列中，消息循环会从此队列中取出消息并分发到各窗口处理
如： WM_PAINT, WM_TIMER, WM_CREATE, WM_QUIT, 以及鼠标， 键盘消息等。
其中， WM_PAINT, WM_TIMER只有在队列中没有其他消息的时候才会被处理，
WM_PAINT消息还会被合并以提高效率。其他所有消息以先进先出（FIFO）的方式被处理。

(2) 非队列消息(NonQueued Messages)

消息会绕过系统消息队列和线程消息队列，直接发送到窗口过程进行处理
如： WM_ACTIVATE, WM_SETFOCUS, WM_SETCURSOR, WM_WINDOWPOSCHANGED

Windows系统的整个消息系统分为**3**个层级：

- ① Windows内核的系统消息队列
- ② App的UI线程消息队列
- ③ 处理消息的窗体对象

Windows内核维护着一个全局的系统消息队列；按照线程的不同，系统消息队列中的消息会分发到应用程序的UI线程的消息队列中；

应用程序的每一个UI线程都有自己的消息循环，会不停地从自己的消息队列取出消息，并发送给Windows窗体对象；

每一个窗体对象都使用窗体过程函数（**WindowProc**）来处理接收到的各种消息。

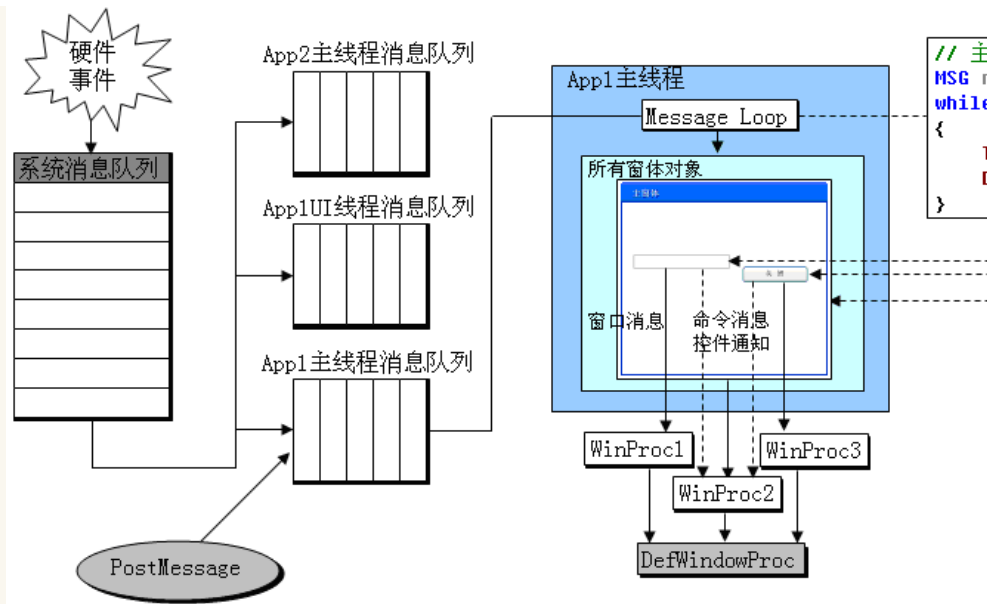
```
1 LRESULT CALLBACK WindowProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
2 {
3     PAINTSTRUCT ps;
4     HDC hdc;
5
6     switch (message)
7     {
8     case WM_COMMAND:
9         break;
10    case WM_PAINT:
11        hdc = BeginPaint(hWnd, &ps);
12        // TODO: 在此添加任意绘图代码...
13        EndPaint(hWnd, &ps);
14        break;
15    case WM_DESTROY:
16        PostQuitMessage(0);
17        break;
18    default:
19        return DefWindowProc(hWnd, message, wParam, lParam);
20    }
21    return 0;
22 }
```

需要的话，在WindowProc中，可以用::GetMessageTime获取当前消息产生的时间，
用::GetMessagePos获取当前消息产生时鼠标光标所在的位置。

Windows-服务(24)
Windows-核心编程(8)
Windows-驱动编程(6)
Windows-网络编程(6)
Windows-系统(22)
编程理解(29)
创业起居注(54)
工具(14)
华为(14)
坚持到底(46)
经典资料(30)
雷军(21)
人生计划(28)
设计模式(1)
时间沙漏(25)
算法(12)
吾日三省(15)
项目管理与架构(15)
行业理解(10)
学习方法(29)
游戏(4)
云计算-学习(2)
赵伟国(7)
正则表达式(3)
最喜欢的(32)

Post Archives

2016/6 (294)
2016/5 (179)
2016/4 (321)
2016/3 (306)
2016/2 (272)
2016/1 (145)
2015/12 (249)
2015/11 (274)
2015/10 (213)
2015/9 (96)
2015/8 (171)
2015/7 (40)
2015/6 (43)
2015/5 (38)
2015/4 (56)
2015/3 (42)
2015/2 (29)
2015/1 (41)
2014/12 (45)
2014/11 (35)
2014/10 (34)
2014/9 (24)
2014/8 (27)
2014/7 (22)
2014/6 (3)
2014/5 (27)
2014/4 (32)
2014/3 (12)
2014/2 (6)
2014/1 (8)
2013/12 (23)
2013/11 (1)
2013/10 (2)
2013/9 (13)
2013/8 (29)
2013/7 (15)
2013/6 (16)
2013/5 (25)
2013/4 (1)
2013/3 (5)
2013/2 (7)
2013/1 (4)
2012/12 (1)
2012/10 (2)
2012/8 (5)
2012/6 (1)
2012/5 (2)
2012/4 (4)
2012/3 (3)



(1) 各个窗口消息由各个窗体（或控件）自身的**WindowProc**（虚函数）接收并处理。

(2) **WM_COMMAND**命令消息统一由当前活动主窗口的**WindowProc**接收，经过绕行后，可被其他的**CCmdTarget**对象处理。

(3) **WM_COMMAND**控件通知统一由子窗口（控件）的父窗口的**WindowProc**接收并处理，也可以进行绕行被其他的**CCmdTarget**对象处理。

（例如：CFormView具备接受WM_COMMAND控件通知的条件，又具备把WM_COMMAND消息派发给关联文档对象处理的能力，

所以给CFormView的WM_COMMAND控件通知是可以让文档对象处理的。）

另外，**WM_COMMAND**控制通知会先调用ReflectLastMsg反射通知子窗口（控件），如果子窗口（控件）处理了该消息并返回TRUE，则消息会停止分发；

否则，会继续调用OnCmdMsg进行命令发送（如同WM_COMMAND命令消息一样）。

注：WM_COMMAND命令消息与WM_COMMAND控件通知的相似之处：

WM_COMMAND命令消息和WM_COMMAND控制通知都是由WindowProc给OnCommand处理，OnCommand通过wParam和lParam参数区分是命令消息或通知消息，然后送给OnCmdMsg处理。事实上，BN_CLICKED控件通知消息的处理和WM_COMMAND命令消息的处理完全一样。因为该消息的通知代码是0，ON_BN_CLICKED(id, memberfunction)和ON_COMMAND(id, memberfunction)是等同的。

（4）**WM_NOTIFY**消息只是对**WM_COMMAND**控件通知进行了扩展，与WM_COMMAND控件通知具有相同的特点。

SendMessage与PostMessage

PostMessage 把消息投递到消息队列后，立即返回；

SendMessage把消息直接送到窗口过程处理，处理完才返回。

GetMessage与PeekMessage

GetMessage 有消息且该消息不为WM_QUIT，返回TRUE。

有消息且该消息为WM_QUIT，返回FALSE。

没有消息时，挂起该UI线程，控制权交还给系统。

PeekMessage 有消息返回TRUE，如果没有消息返回FALSE；不会阻塞。

是否从消息队列中删除此消息（PM_REMOVE），由函数参数来指定。

要想在没有消息时做一些工作，就必须使用PeekMessage来抓取消息，以便在没有消息时，能在OnIdle中执行空闲操作（如下）：

```
1 while (TRUE)
2 {
3     if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
4     {
5         if (msg.message == WM_QUIT)
6             break;
7         TranslateMessage(&msg);
```

2012/2 (8)
2012/1 (3)
2011/11 (1)
2011/10 (3)

Article Categories

Delphi-BPL(1)

Recent Comments

1. Re:Delphi组件开发-在窗体标题栏添加按钮（使用MakeObjectInstance(NewWndProc)，并处理好多消息）
XE运行的时候显示不出来
visible:=True也不行
--我拿青春赌明天
2. Re:windows消息机制（MFC）
对初学者了解MFC来说很有帮助！满赞
--于得水007
3. Re:Qt 编程指南（一本全中文的书，尤其QString和QByteArray讲的不错）
因为作者太忙了，貌似没空写了。
--I-M
4. Re:模拟QQ系统设置面板实现功能
有个问题就是，如果QScrollArea里面的东西太少，撑不到滚动。那么
setSliderPosition无论怎么设置，都没法滚动。这个问题如何解决呢？
--milu200
5. Re:阿里云消息应用（慢慢研究）
楼主，研究心得分享一下啊，最近也打算往云上迁
--hustercp

Top Posts

1. JQuery动态隐藏和显示DIV(175584)
2. JS把内容动态插入到DIV(24622)
3. 终于理解了什么是LGPL(17314)
4. UTF8最好不要带BOM，附许多经典评论(15903)
5. jquery隐藏按钮(13109)

推荐排行榜

1. JQuery动态隐藏和显示DIV(6)
2. 计算机视觉牛人博客和代码汇总（全）(5)
3. 终于理解了什么是LGPL(3)
4. UTF8最好不要带BOM，附许多经典评论(2)
5. 站在巨人的肩膀上，C++开源库大全(2)

```
8         DispatchMessage (&msg);
9     }
10    else
11    {
12        OnIdle ();
13    }
14 }
```



例如：MFC使用OnIdle函数来清理一些临时对象及未使用的动态链接库。
只有在OnIdle返回之后程序才能继续处理用户的输入，因此不应在OnIdle进行较长的任务。

MFC消息处理

在CWnd中，MFC使用OnWndMsg来分别处理各类消息：

如果是WM_COMMAND消息，交给OnCommand处理；然后返回。

如果是WM_NOTIFY消息，交给OnNotify处理；然后返回。

如果是WM_ACTIVATE消息，先交给_AfxHandleActivate处理，再继续下面的处理。

如果是WM_SETCURSOR消息，先交给_AfxHandleSetCursor处理，然后返回。

如果是其他的窗口消息（包括WM_ACTIVATE消息），则

- 首先在消息缓冲池（一个hash表，用于加快消息处理函数的查找）进行消息匹配，若匹配成功，则调用相应的消息处理函数；
- 若不成功，则在消息目标的消息映射数组中进行查找匹配，看它是否能处理当前消息。
- 如果消息目标处理了该消息，则会匹配到消息处理函数，调用它进行处理；

否则，该消息没有被应用程序处理，OnWndMsg返回FALSE。

MFC消息映射

消息映射实际是MFC内建的一个消息分派机制。

把MFC中的宏进行展开（如下），可以得到消息映射表整个全貌。

```

struct AFX_MSGMAP_ENTRY
{
    UINT nMessage;    // windows message
    UINT nCode;       // control code or WM_NOTIFY code
    UINT nID;         // control ID (or 0 for windows messages)
    UINT nLastID;     // used for entries specifying a range of control id's
    UINT_PTR nSig;     // signature type (action) or pointer to message #
    AFX_PMSG pfn;     // routine to call (or special value)
};

typedef void (AFX_MSG_CALL CCmdTarget::*AFX_PMSG)(void);

struct AFX_MSGMAP
{
    const AFX_MSGMAP* (PASCAL* pfnGetBaseMap)();
    const AFX_MSGMAP_ENTRY* lpEntries;
};

#define DECLARE_MESSAGE_MAP() \
protected: \
    static const AFX_MSGMAP* PASCAL GetThisMessageMap(); \
    virtual const AFX_MSGMAP* GetMessageMap() const; \

BEGIN_MESSAGE_MAP(CTestView, CView)
    ON_COMMAND(ID_FILE_OPEN, &CTestView::OnFileOpen)
    ON_WM_PAINT()
END_MESSAGE_MAP()

#define BEGIN_MESSAGE_MAP(theClass, baseClass) \
const AFX_MSGMAP* theClass::GetMessageMap() const \
{ return GetThisMessageMap(); } \
const AFX_MSGMAP* PASCAL theClass::GetThisMessageMap() \
{ \
    typedef theClass ThisClass; \
    typedef baseClass TheBaseClass; \
    static const AFX_MSGMAP_ENTRY _messageEntries[] = \
    { \

#define ON_COMMAND(id, memberFxn) \
    { WM_COMMAND, CN_COMMAND, (WORD)id, (WORD)id, AfxSigCmd_v, \
      static_cast<AFX_PMSG> (memberFxn) }, \
    // ON_COMMAND(id, OnBar) is the same as \
    // ON_CONTROL(0, id, OnBar) or ON_BN_CLICKED(0, id, OnBar)

#define ON_WM_PAINT() \
    { WM_PAINT, 0, 0, 0, AfxSig_vv, \
      (AFX_PMSG)(AFX_PMSGW) \
      (static_cast< void (AFX_MSG_CALL CWnd::*)(void) > ( &ThisClass ::

#define END_MESSAGE_MAP() \
    { 0, 0, 0, 0, AfxSig_end, (AFX_PMSG)0 } \
}; \
static const AFX_MSGMAP messageMap = \
{ &TheBaseClass::GetThisMessageMap, &_messageEntries[0] }; \
return &messageMap; \

} \

typedef void (AFX_MSG_CALL CWnd::*AFX_PMSGW)(void);
// like 'AFX_PMSG' but for CWnd derived classes only

```

注：GetMessageMap为虚函数。

{0, 0, 0, 0, AfxSig_end, (AFX_PMSG)0}：对象消息映射表的结束标识

窗口消息只能由CWnd对象来处理，采用向基类直线上溯的方式，来查找对应的消息响应函数进行处理。

一旦找到消息响应函数（若有返回值且为TRUE），就停止上溯。因此，我们经常会看到这样的代码：

```

void CTreeView::OnSize(UINT nType, int cx, int cy)
{
    CView::OnSize(nType, cx, cy);

    // TODO: Add your message handler code here
}

BOOL CTreeView::OnEraseBkgnd(CDC* pDC)
{
    // TODO: Add your message handler code here and/or call default

    return CView::OnEraseBkgnd(pDC);
}

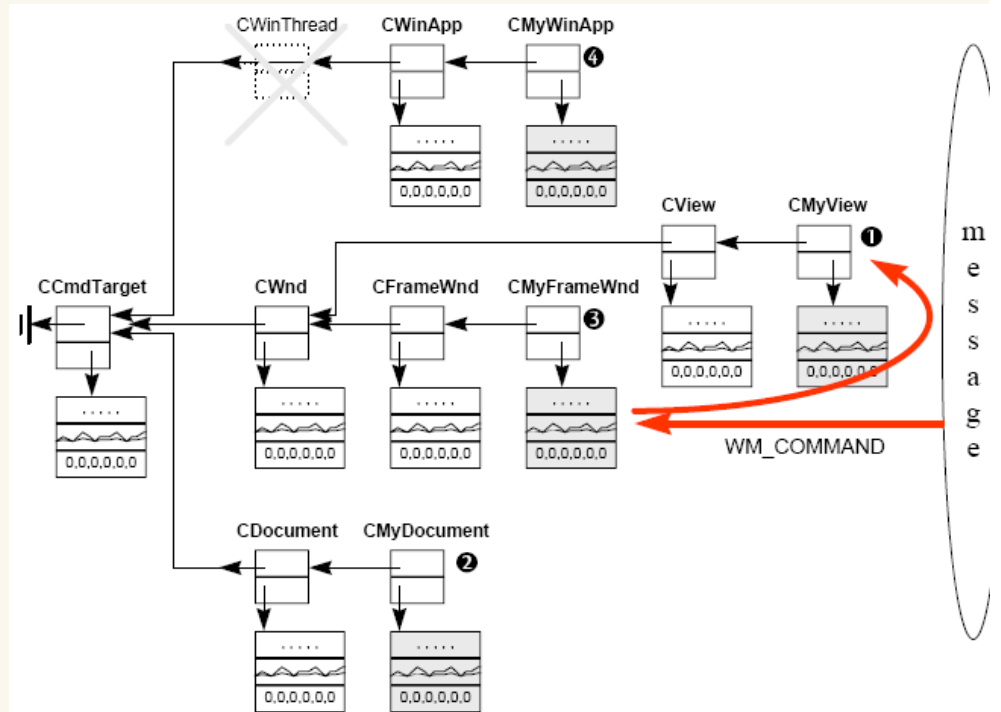
```

增加一个消息处理函数来写我们的逻辑时，MFC ClassWizard会在该函数之前或之后显示调用其基类对应的函数，保证基类中逻辑被执行。

命令消息可由**CCmdTarget**对象接收并处理（OnCmdMsg为虚函数），除了向基类直线上溯方式外，还有命令绕行机制（要防止形成圈，死循环）。

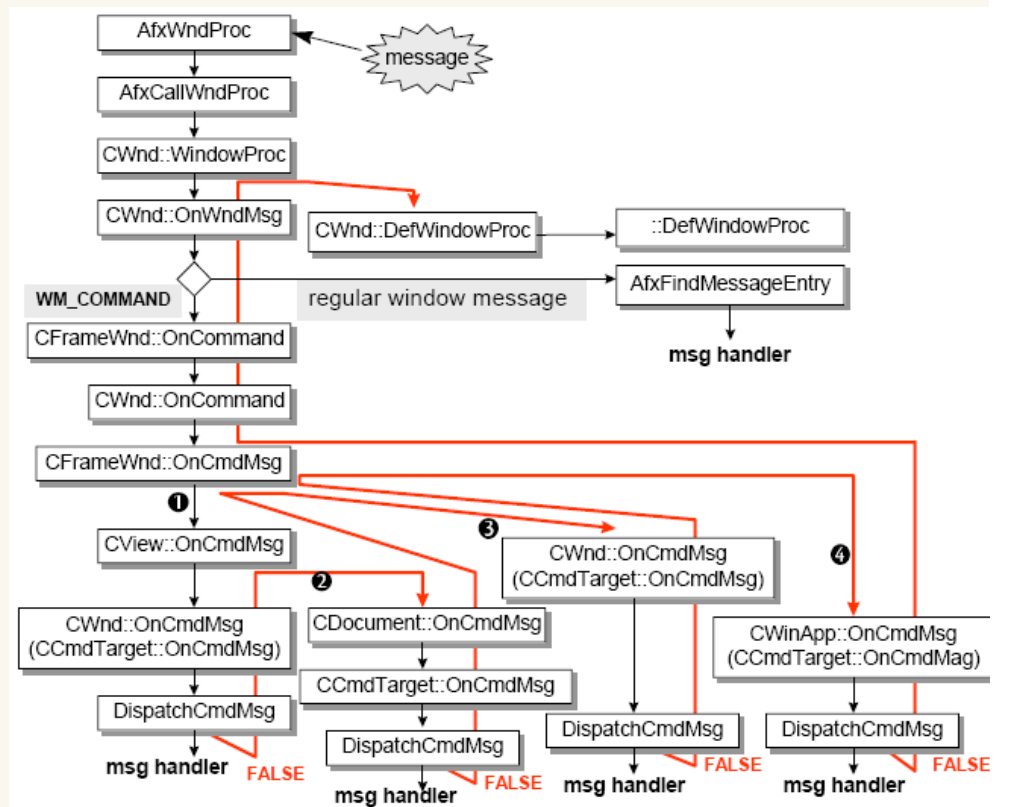
在某种程度上，控制通知消息由窗口对象处理是一种习惯和约定。然而，控件通知消息也是可以有**CCmdTarget**对象接收并处理，并进行命令绕行的。

下图为**MFC**经典单文档视图框架的命令消息绕行路线：



FrameWnd 窗口收到命令消息后的四个尝试路径（图中的①②③④即绕行的优先次序）

函数调用过程如下（如果没有任何对象处理该条WM_COMMAND消息，最后会被::DefWindowProc处理）。



非模态对话框的消息处理

```
1 static CAboutDlg aboutDlg;
2 aboutDlg.Create(IDD_ABOUTBOX, this);
3 aboutDlg.ShowWindow(SW_SHOW);
```


应用程序只有一个消息循环。

对于窗口消息，非模态对话框（及其子控件）与父窗口（及其子控件）都是用自身的WindowProc函数接收并处理，互不干扰。

对于命令消息，由当前活动主窗口的WindowProc接收（例如：当前活动主窗口为非模态对话框，则命令消息会被非模态对话框接收）。

可以在当前活动主窗口的OnCmdMsg中做命令绕行，使得其他的CCmdTarget对象也可以处理命令消息。

对于控件通知，由其父窗口的WindowProc接收并处理，一般不进行命令绕行被其他的CCmdTarget对象处理。

模态对话框的消息处理

```
1 CAboutDlg aboutDlg;  
2 aboutDlg.DoModal();
```

(1) 模态对话框弹出来后，首先会让父窗口失效，使其不能接受用户的输入（键盘鼠标消息）。

```
1 EnableWindow(hwndParent, FALSE);
```

(2) 父窗口消息循环被阻塞（会卡在DoModal处，等待返回），由模态对话框的消息循环来接管（因此整个程序不会卡住）。

接管后，模态对话框的消息循环仍然会将属于父窗口及其子控件的窗口消息（不包括键盘鼠标相关的窗口消息）发送给它们各自的WindowProc窗口函数，进行响应处理。

(3) 模态对话框销毁时（点击IDOK或IDCANCEL），父窗口消息循环重新激活，继续DoModal后的逻辑。

激活后，父窗口有可以重新接受用户的输入（键盘鼠标消息）。

```
1 EnableWindow(hwndParent, TRUE);
```

从上面的过程中，我们可以得到如下结论：

对于窗口消息，模态对话框主窗口（及其子控件）与父窗口（及其子控件）都是用自身的WindowProc函数接收并处理，互不干扰。

只是父窗口（及其子控件）无法接受到键盘鼠标消息相关的窗口消息。

对于命令消息，由模态对话框主窗口的WindowProc接收。可以在模态对话框主窗口的OnCmdMsg中做命令绕行，使得其他的CCmdTarget对象也可以处理命令消息。

对于控件通知，由其父窗口的WindowProc接收并处理，一般不进行命令绕行被其他的CCmdTarget对象处理。

参考

《深入浅出MFC》- 侯捷

[《MFC教程》- 消息映射的实现](#)

<http://blog.csdn.net/kongfuxionghao/article/details/35882533>

分类: [VC++ MFC](#), [Delphi-消息大全](#)



 [findumars](#)
[关注 - 55](#)
[粉丝 - 112](#)
[+加关注](#)

[1](#) [0](#)
[推荐](#) [反对](#)

(请您对文章做出评价)

« 上一篇: [鼠标消息与键盘消息（文章很长很全面）](#)

» 下一篇: [C++ Development Library](#)


posted @ 2014-08-31 23:42 findumars Views(3588) Comments(3) Edit 收藏

Post Comment

#1楼 2015-06-14 11:36 | 黑桃七

“
图是楼主自己画的吗？

支持(0) 反对(0)

#2楼 2016-04-17 14:08 | fs2014 

“

看了很多MFC相关的帖子，楼主写的很明白，谢谢

支持(0) 反对(0)


#3楼 2016-06-16 10:40 | 于得水007 

“

对初学者了解MFC来说很有帮助！满赞

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

 注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

最新**IT**新闻：

- 传统游戏主机将死，未来主机将效仿智能手机升级模式
 - 财富：小米配得上450亿美元的估值吗？
 - 百度的冬天：曾梦想成伟大公司 却为何陷入危机
 - 印度智能手机Freedom 251下周发货 价格仅相当于一杯咖啡
 - 谷歌获得新专利 积极研发仓库机器人技术
- » 更多新闻...

最新知识库文章：

- 我听到过的最精彩的一个软件纠错故事
 - 如何避免软件工程中最昂贵错误的发生
 - 拒绝传统，看 Facebook 如何以三大法宝化茧成蝶
 - 当我们谈论极简设计，我们在谈些什么
 - 用户体验设计遇见色彩情感
- » 更多知识库文章...



Copyright ©2016 findumars