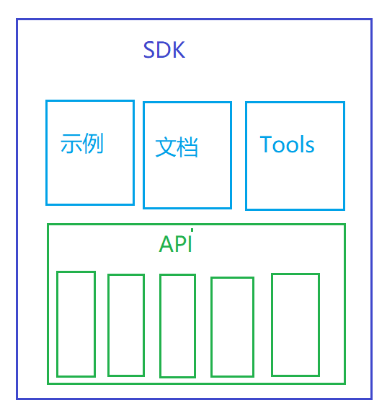
API和SDK

API(Application Program Interface)，应用程序编程接口，可以理解为底层模块给上层模块提供的一系列函数，供上层模块进行调用。当底层模块比较庞大时（比如Windows 操作系统），会分类以模块的形式提供API

SDK(Software Development Kit), 软件开发工具包，就是将上述的所有API，以及和API相关的一些工具等组合在一起，提供给用户进行安装的Package



控制台程序和图形界面程序

控制台 console，就是字符界面的，象”Command Prompt”

图形界面 GUI, Graphical User Interface，就是图形界面，可以支持鼠标等设备操作，目前绝大多数客户端程序都是图形界面的

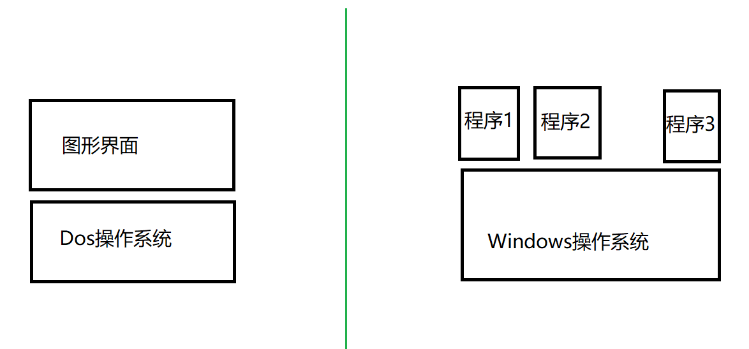
图形界面是需要操作系统OS，Operation System来支持的，象Windows, Mac，Android等都是支持图形界面的OS

象Linux, Unix主要是支持字符界面的OS，因为它们一般是作为服务端程序在后台运行，不面对客户。（当前也有公司基于Linux基础之上开发图形界面的，但用户量不大）

[以下以Windows为例而讨论]

在Windows95之前，是在Dos（字符界面）的基础之上，构建图形界面

在Windows95之后，MS开始从底层就支持图形界面，而Command Prompt只是一个字符界面的程序



在新的操作系统下，用户开发的程序和MS提供的程序，从地位上来说是完全一样的，只是开发者主体不同而已，甚至有人开发了功能更强的程序直接替掉MS提供的程序

消息

Windows操作系统的核心是消息 Message，称为Windows Message，以一系列是WM\_开头的预定义好的整型数值

A picture containing text

Description automatically generated

系统也有预留的可以扩展的字段，让用户自行定义，但格式是完全一样的

消息队列

每一个图形界面的应用程序都有自己的一个消息队列，Application Message Queue，同时由于象鼠标键盘等共享设备，也会产生消息，但是它是由系统进行管理的，所以另有一个所有程序共享的系统消息队列，System Message Queue

消息循环

学习过编程语言的都知道循环语句，有while, do…while, for等（Java, C/C++, Python都有的）

在Windows图形界面程序中，也有一个这样的循环语句，只不过，循环体内处理的对象是Windows Message，所以称之为消息循环

其中的msg的定义为 *MSG* msg;

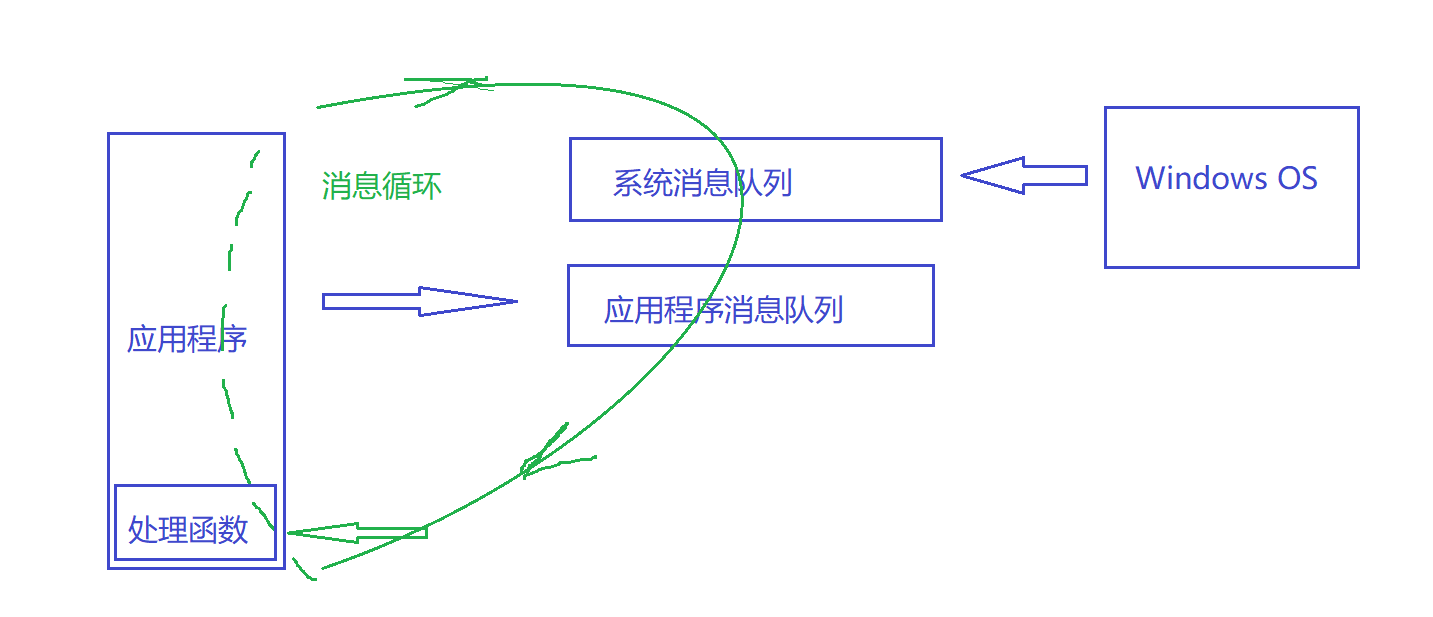
就是一个Windows Message Object

Graphical user interface, text, application

Description automatically generated

除非收到WM\_QUIT消息，GetMessage一直返回 True 值，就会一直循环下去。只有收到WM\_QUIT后才会返回False，此时while循环才退出，程序结束

消息处理



当用户在程序里进行操作（点击按钮或菜单）或者键盘鼠标输入时，Windows OS并不是直接调用处理函数进行处理，而是先生成对应的消息，放入消息队列中（这是OS的操作），用户程序通过消息循环可以从消息队列中获取到消息，然后调用对应的处理函数（这是用户程序的操作）

控制台程序

入口函数是main

int main()

整个程序比较简单，按流程顺序逐步执行，

Rule:

1. 函数申明定义在一个.h 文件中
2. 函数实现代码在一个.cpp 文件中

3，.h文件的第一句应该是 #pragma once

相关的设置在属性页 Properties，基本介绍

Graphical user interface, text, application

Description automatically generated

图形界面程序 (这个是重点，所以的底层逻辑都在这里)

1. 项目生成向导
2. 资源文件，包括资源ID定义Resource.h以及资源描述文件\*.rc
3. 主函数入口文件

int *APIENTRY* *wWinMain*(… …)

整个程序是有一套统一的框架结构和调用顺序

1. 注册窗口类
   1. 重点是窗口类名，和窗口过程
2. 创建主窗口
   1. 重点是窗口类名，因为通过窗口类名，就可以关联到对应的窗口过程
3. 显示主窗口
4. 主窗口消息循环

5，在窗口过程中进行消息的处理

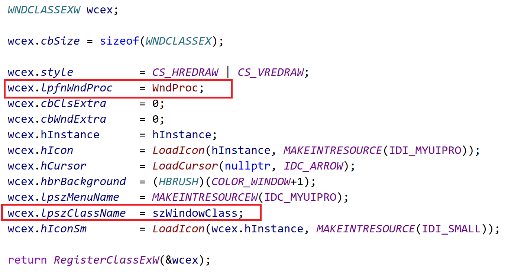
6，主窗口过程一定要对WM\_DESTROY进行处理，并调用 *PostQuitMessage*(0);

对相关代码的解释说明

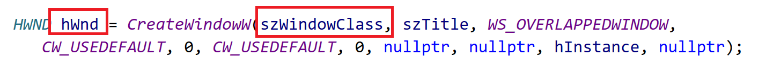
总结：

一、Windows程序设计基本流程

1. 入口是 *wWinMain*
2. 注册窗口类 *RegisterClassExW*(&wcex)



1. 创建窗口对象 *CreateWindowW*(…)



1. 显示窗口对象 *ShowWindow*(hWnd, nCmdShow)

Graphical user interface, text

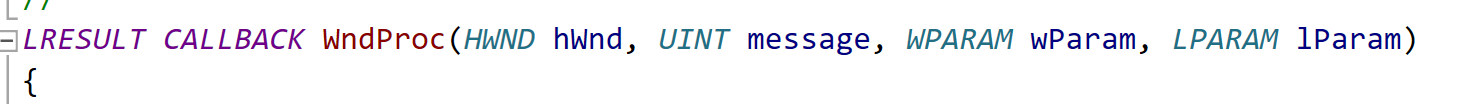
Description automatically generated

1. 进入消息循环 while (*GetMessage*(&msg, nullptr, 0, 0))

Text, letter

Description automatically generated

1. 在窗口过程中处理窗口消息 *LRESULT* *CALLBACK* WndProc(…)



1. 退出一定调用 *PostQuitMessage*(0);

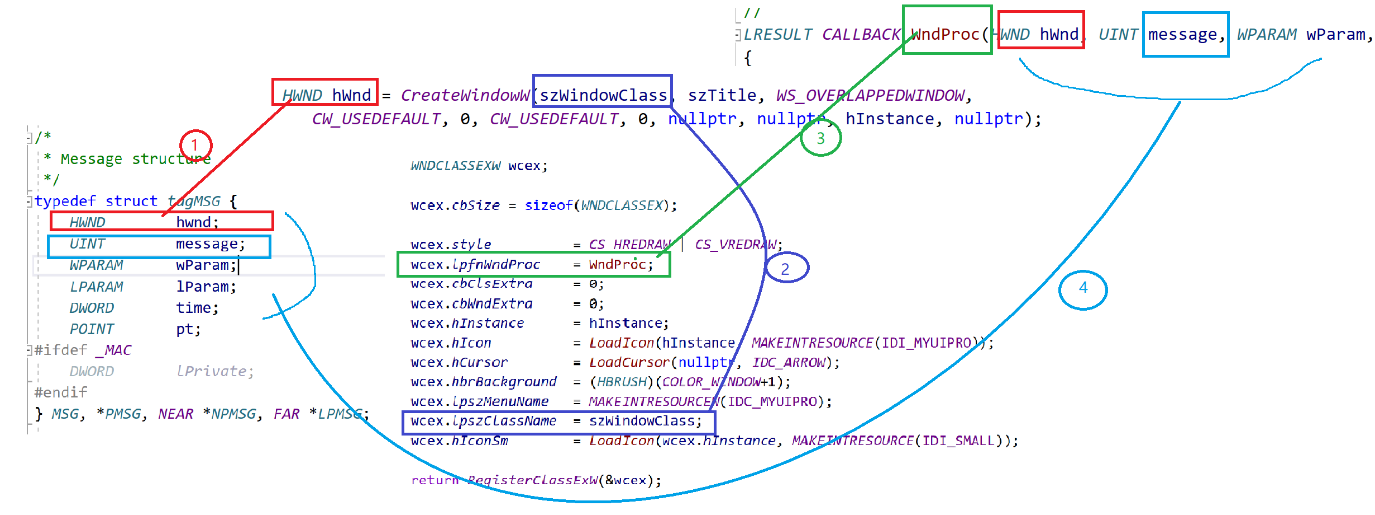
A picture containing logo

Description automatically generated

二、Windows程序消息派发逻辑

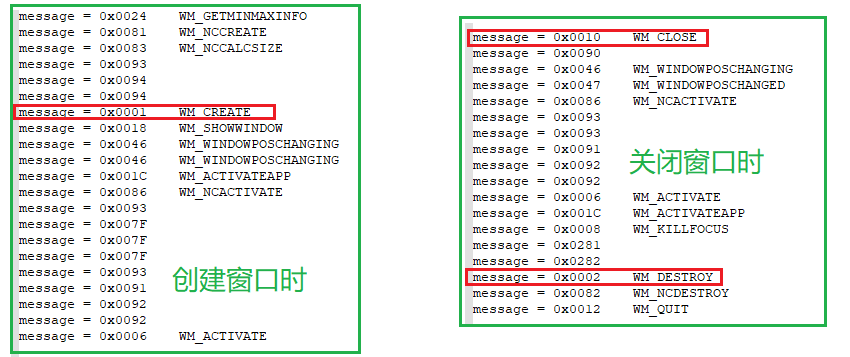
1. 从消息队列中获取的消息Message，是一个结构体，除了跟消息相关的参数外，里面还有一个窗口句柄hWnd
2. 根据窗口句柄hWnd，可以找到该窗口Window。因为窗口句柄是描述窗口的唯一标识符，相当于一个人的身份证号
3. 所有的窗口在创建时都会指定一个窗口类名Window Class Name。相当于一个人出生报户口生成身份证时，你得告诉你的出生地。这样根据身份证号就知道出生在哪里，同样根据窗口可以找到属于哪一种窗口类
4. 而窗口类在注册时会关联一个窗口过程 Window Process
5. 顺着上面的逻辑，我们对每一个消息，都可以找到唯一对应的窗口过程 （其实就是一个全局的函数，函数的名称就表示该函数在内存中的地址）
6. 找到窗口过程后，就调用该函数，同时将消息结构体中的参数作为实参传入函数中

这就是Windows Message Dispatch Logic

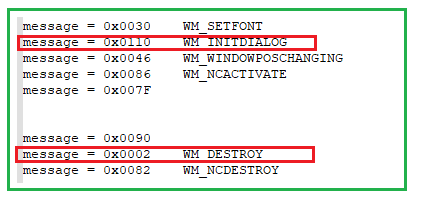


三、在一个窗口创建的过程中，大概有哪些消息？

普通窗口Window



对话框Dialog



四，关于对话框

对话框Dialog是一种特殊类型的窗口Window，是需要提供资源Resource

对话框的显示又分为模态和非模态。

|  |  |  |  |
| --- | --- | --- | --- |
|  | Window | Model Dialog | Modeless Dialog |
| 创建 | *CreateWindow* | *DialogBox* | *CreateDialog* |
| 销毁 | *DestroyWindow* | *EndDialog* | *DestroyWindow* |
| 消息循环 | 外部消息循环 | 内部消息循环 | 外部消息循环 |
| 资源 | 不用 | 需要 | 需要 |
| 创建消息 | *WM\_CREATE* | *WM\_INITDIALOG* | *WM\_INITDIALOG* |
| 销毁消息 | *WM\_DESTROY* | *WM\_DESTROY* | *WM\_DESTROY* |
|  |  |  |  |

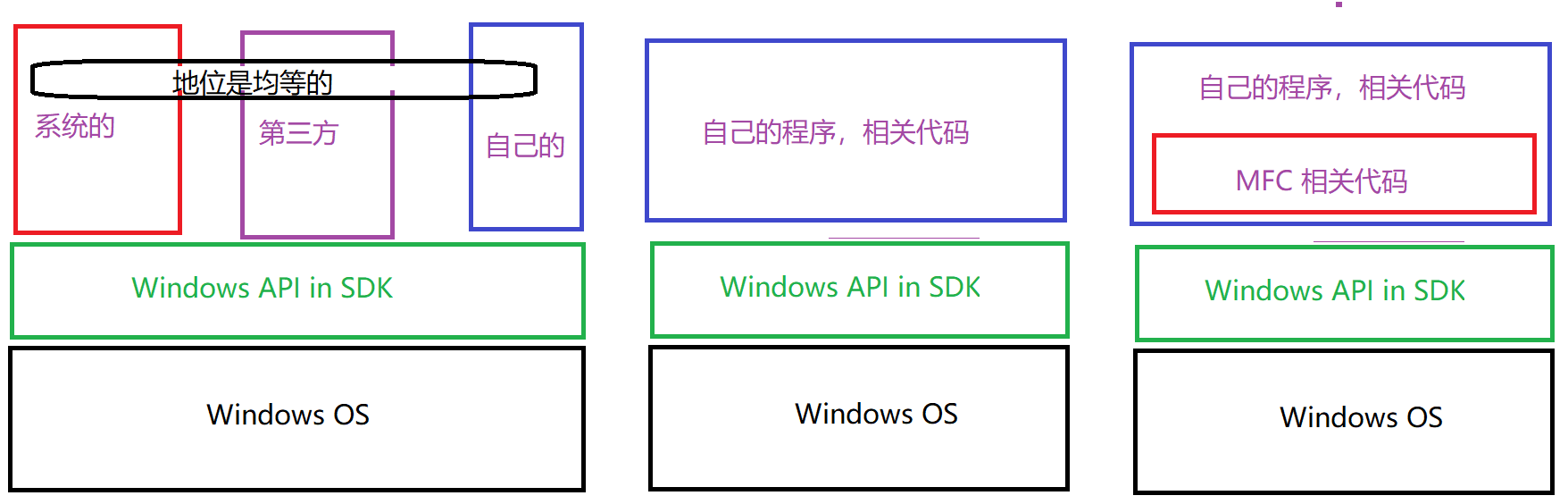
基于MFC的图形界面程序

微软基础类库（英语：Microsoft Foundation Classes，简称MFC）

核心原理就是上一节的图形界面程序（其实所有的Windows图形界面程序都是这一套标准框架）

但是它是基于类的思想和封装的思想，把一些用户不是必须关心的琐事封装到后台，用户只需要关心感兴趣的内容：

比如窗口类的风格，窗口消息以及该消息对应的处理函数等



基于MFC，我们需要编写的代码量会少很多，大部分低层的代码，已经被MFC进行了封装，我们只需要关注跟业务相关的代码

但它所带来的不好地方，就是隐藏了细节，对于初学者来说，搞不清楚Windows程序设计中的一些底层的，关键的逻辑

对MFC程序的理解，必须有一定的基础，包括：

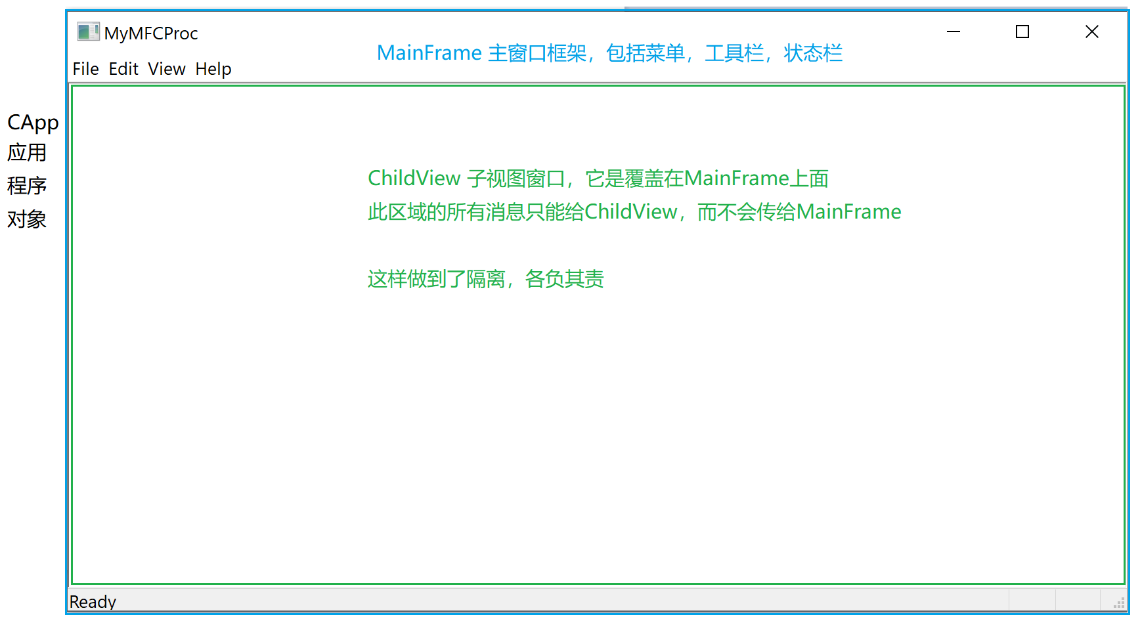
OOP，面向对象的编程思想

Windows消息机制，以及处理过程

底层封装，高层自治

界面设计工具集

代码向导



用MFC做一个基于对话框的小程序，

用工具填加资源，

用向导生成代码

Graphical user interface, application

Description automatically generated

附MFC程序系统框架

Application，程序总控

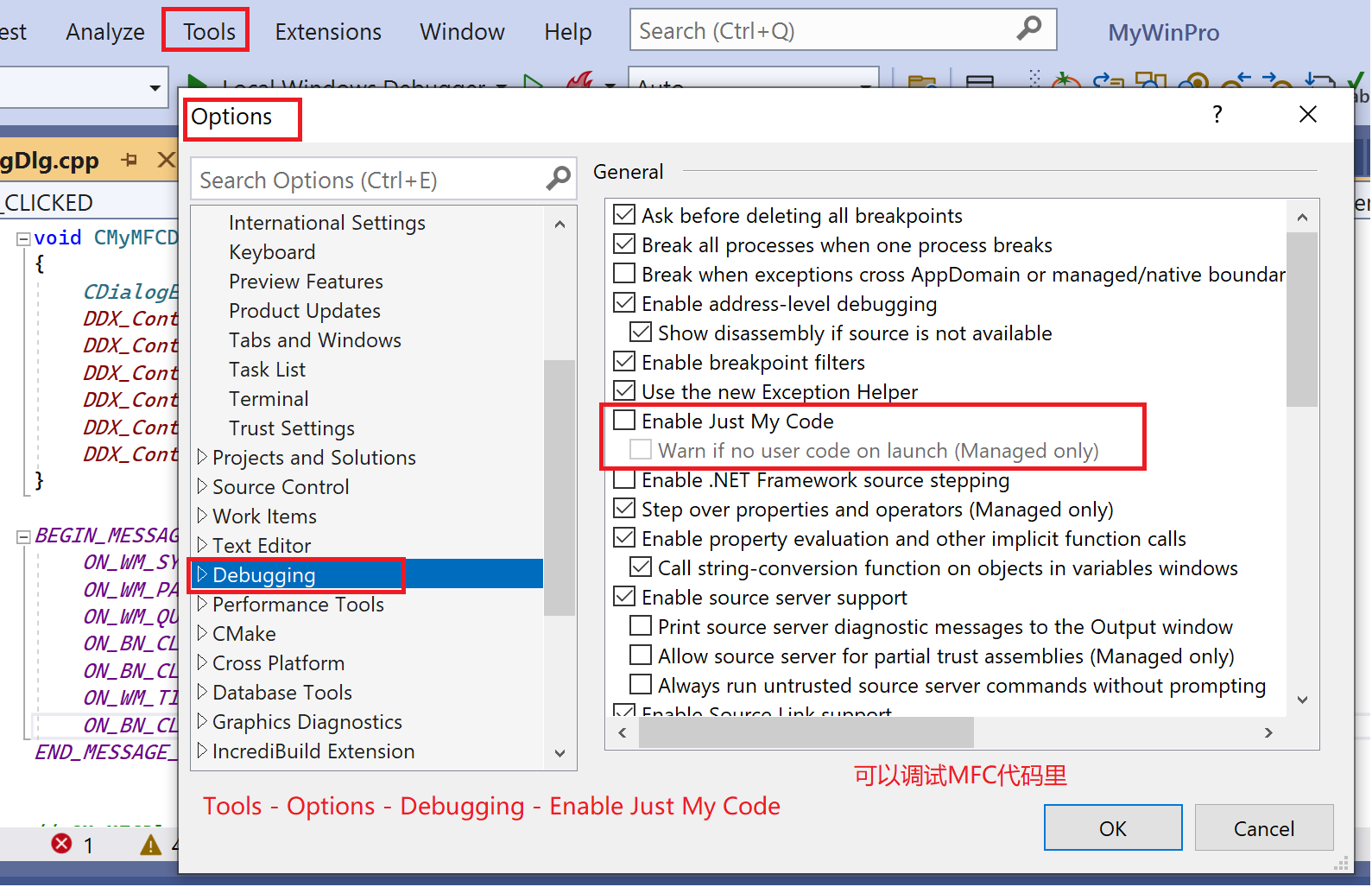
Main Frame，程序外观框架，以及交互界面

View，程序显示区域，它是Main Frame的一个子窗口

Document，View区域所要显示的内容

还有一个把Main Frame, View和Document联系起来的内部整理组件 DocTemplate





自己重载的函数只有少数的几个，会被MFC回调过来 (面向对象中的虚函数机制)

*BOOL* CMyMFCProcApp::InitInstance();

在消息循环之前调用它

int CMyMFCProcApp::ExitInstance();

在消息循环之后调用它

*BOOL* CMainFrame::PreCreateWindow(*CREATESTRUCT*& cs);

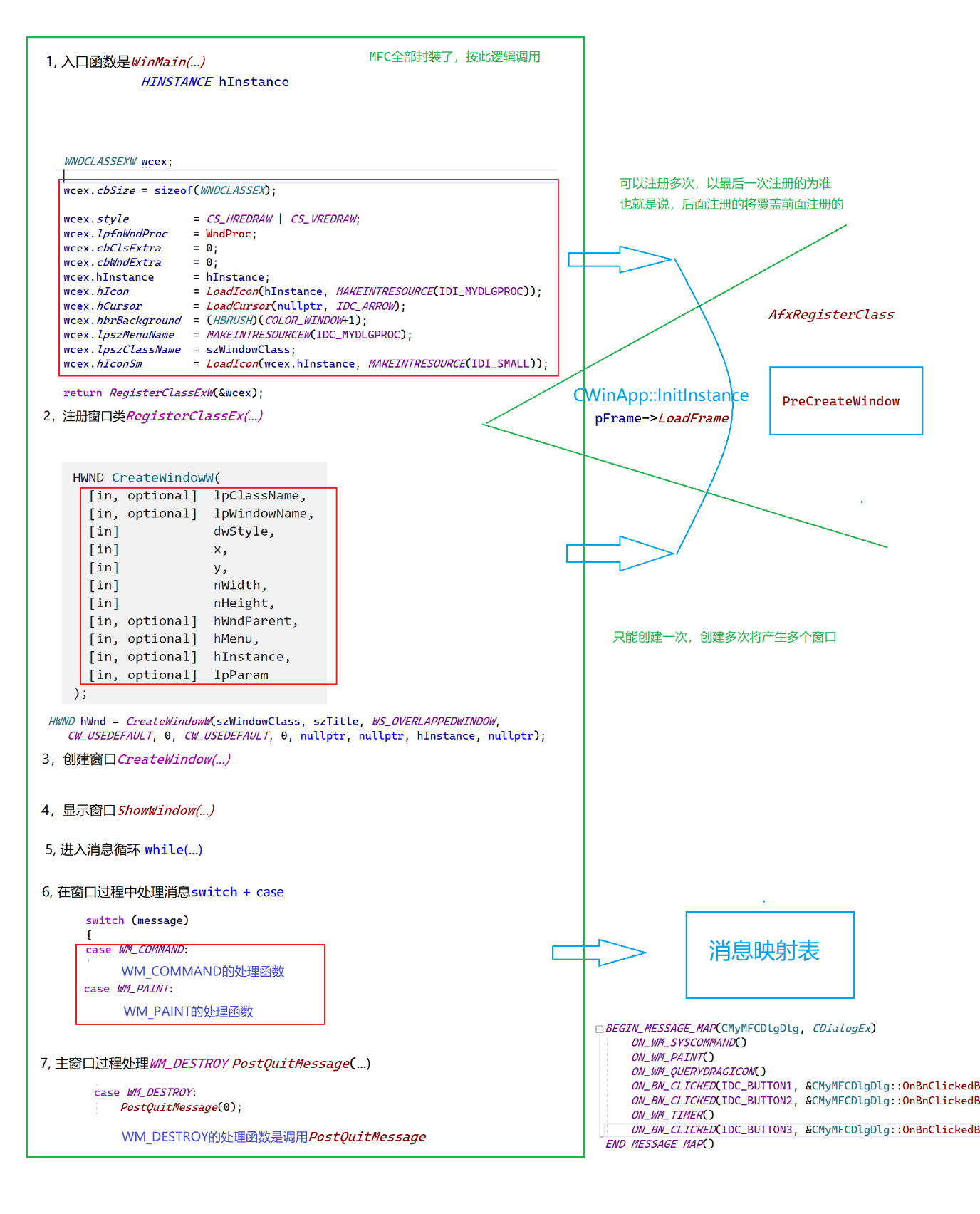
在创建主窗口之前调用它，可以修改或定制主窗口类，以及主窗口风格等

*BOOL* CChildView::PreCreateWindow(*CREATESTRUCT*& cs);

在创建子窗口之前调用它，可以修改或定制子窗口类，以及子窗口风格等

学会使用spy / spy++

以前叫spy，现在扩展其功能，更名为spy++，可以查看窗口所有属性，以及窗口的消息



**MFC命令的路由**

关于WM\_COMMAND，由菜单，按钮或者用户自己触发的命令类消息，有一个特殊的派发过程，会经历所有的类（对象）

virtual *BOOL* OnCmdMsg(*UINT* nID, int nCode, void\* pExtra,

*AFX\_CMDHANDLERINFO*\* pHandlerInfo);

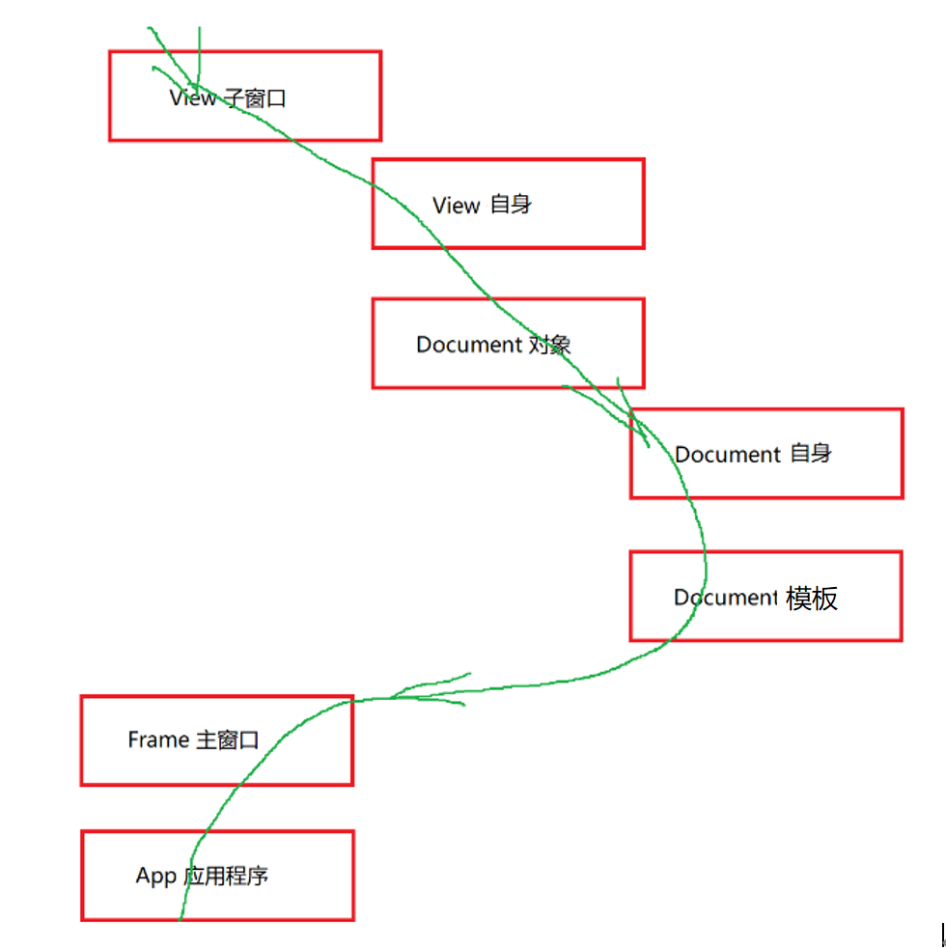
OnCmdMsg( )是在CCmdTarget类中的一个基本函数，主要是用来处理消息的路由的。

Table

Description automatically generated

Text

Description automatically generated



**几篇不错的文章**

**MFC应用程序中处理消息的顺序**

<https://blog.csdn.net/jn_mzh/article/details/4834680>

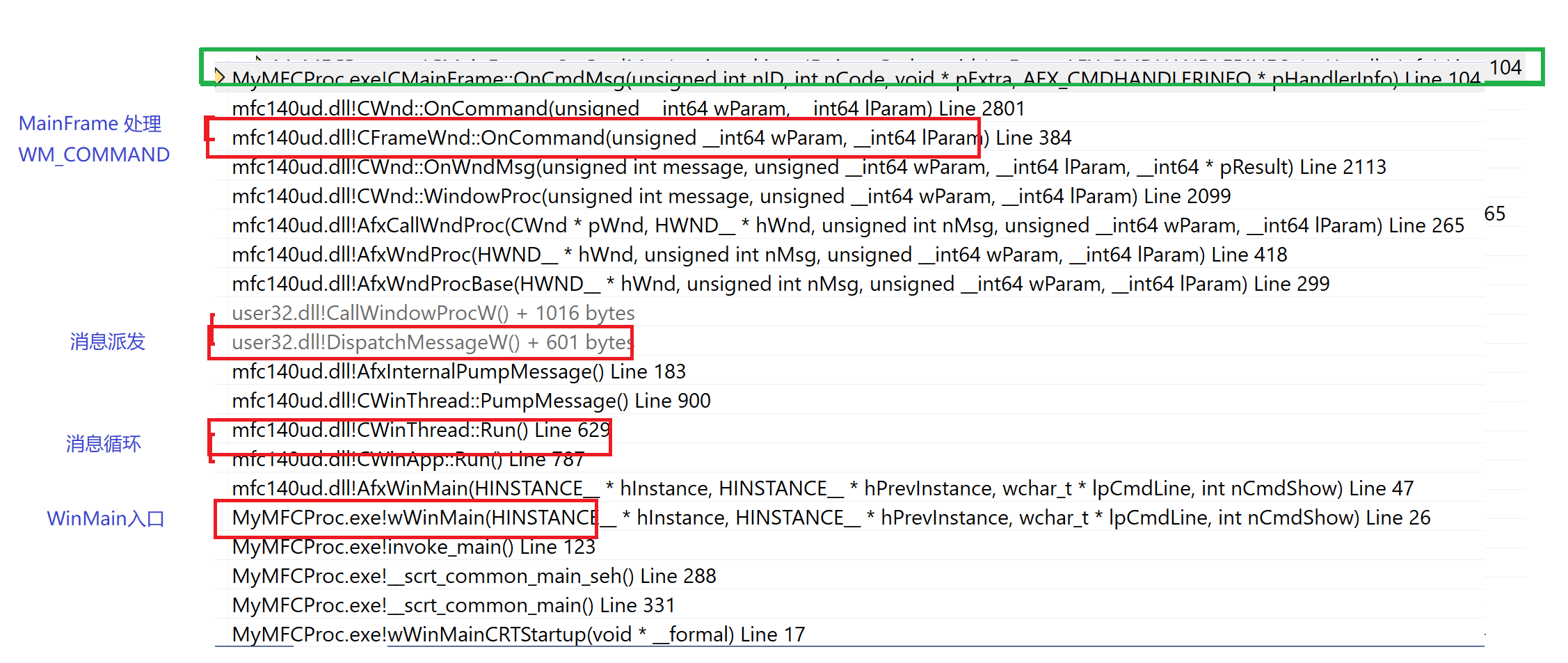
**MFC中的命令传递(Command Routing)**

<https://blog.csdn.net/sanqima/article/details/42339929>

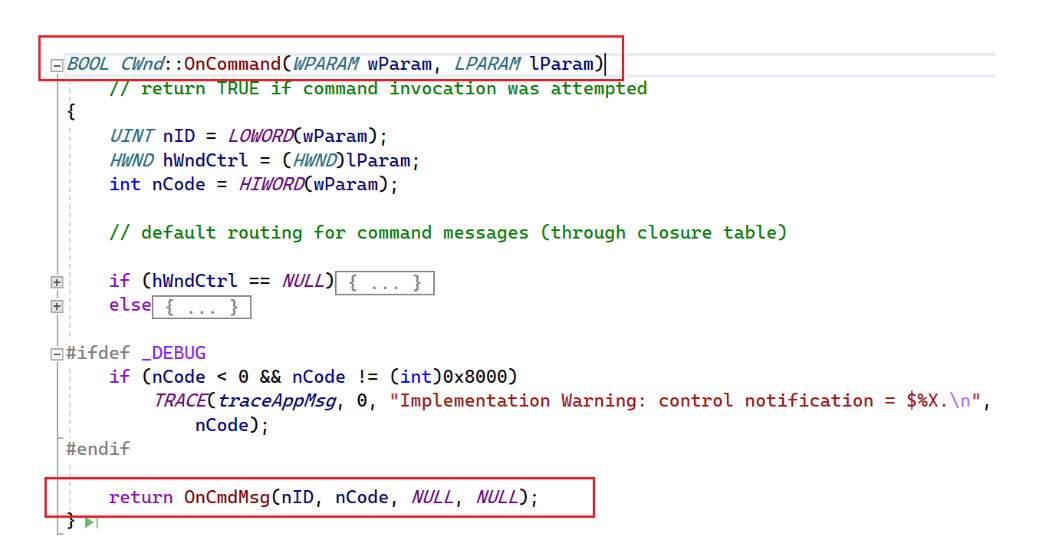
**Windows常用消息大全和Windows消息机制（总结篇）**

<https://blog.csdn.net/qq_45021180/article/details/97949267>

调用栈CallStack



从OnCommand到OnCmdMsg的代码



命令路由的代码

Text

Description automatically generated

