[1. 语法 2](#_Toc430777312)

[1-1. Afx\_msg 声明是否多余 2](#_Toc430777313)

[1-2. C、C++、MFC宏 2](#_Toc430777314)

[1-2-1. #pragma once 2](#_Toc430777315)

[1-3. 字符和字符串 2](#_Toc430777316)

[1-3-1. C++ 定义字符串时，用L标示是ANSI字符还是Unicode字符 2](#_Toc430777317)

[1-3-2. \_T的作用 2](#_Toc430777318)

[1-3-3. c++ 字符类型总结区别wchar\_t,char,WCHAR 3](#_Toc430777319)

[1-3-4. Visual Studio 2013已经不支持非Unicode的工程的编译 3](#_Toc430777320)

[2. MFC结构 3](#_Toc430777321)

[2-1. CDialog 3](#_Toc430777322)

[2-1-1. 头文件和实现文件 3](#_Toc430777323)

[2-1-2. 类里声明了OnOK和OnCancel方法，和自己的MsgMap 3](#_Toc430777324)

[2-1-3. 在自己的类里处理Dialog的IDOK按钮的click事件 4](#_Toc430777325)

[3. Project 5](#_Toc430777326)

[3-1. ListCtrlDemo 5](#_Toc430777327)

# 语法

## Afx\_msg 声明是否多余

afx\_msg void OnPaint();

貌似没啥用。不过查了网上的说明还是留下算了。

你可以在afxwin.h中找到如下的定义  
#ifndef afx\_msg  
#define afx\_msg         // intentional placeholder  
#endif  
就是什么也没有做  
看它的注释 intentinal placeholder 一个有意的占位符  
应该是为将来可能的升级所准备的  
所以 最好不要删除你现有程序中的afx\_msg

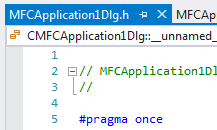
没什么意思.只是定义了这个符号而已. 这个对编译器来说,相当于什么都没有,对于人来说,我们可以看到这样的符号. 对于类向导来说.这个符号才是有意义的.它是一个消息处理函数的前缀. 类向导生成的消息函数,分发函数,事件响应函数都以这个为前缀. 如果去掉了,向导将不能识别。

## C/C++中声明指针变量时星号是靠近变量名还是靠近类型?

## C、C++、MFC宏

[C、C++、MFC宏](Doc/C、C++、MFC宏.pdf)

### #pragma once



In the C and C++ programming languages, #pragma once is a non-standard but widely supported preprocessor directive designed to cause the current source file to be included only once in a single compilation.

## 字符和字符串

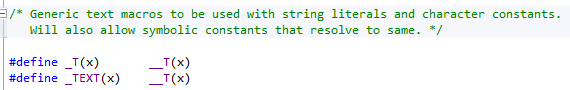
### C++ 定义字符串时，用L标示是ANSI字符还是Unicode字符

this->textBox1->Name = L"textBox1";

It's a wchar\_t literal, for extended character set. [Wikipedia](http://en.wikipedia.org/wiki/Wide_character) has a little discussion on this topic, and c++ examples.

### \_T的作用

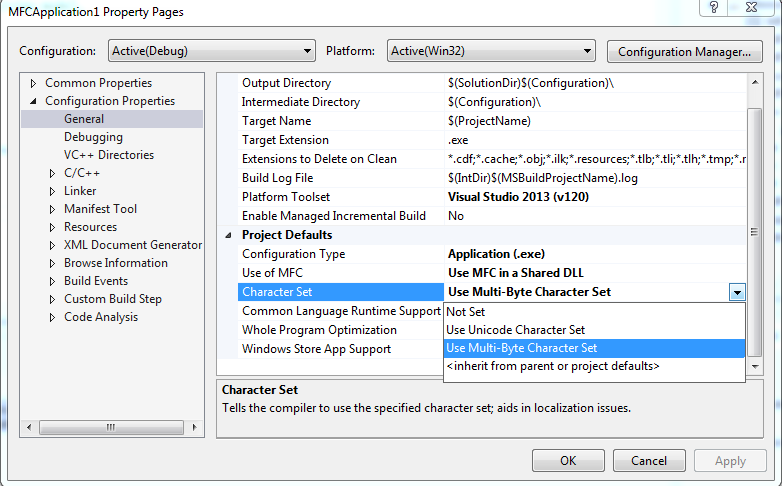
它其实是tchar.h里定义的一个宏



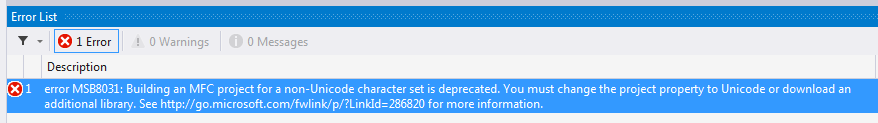
### c++ 字符类型总结区别wchar\_t,char,WCHAR

[c++ 字符类型总结区别wchar\_t,char,WCHAR](Doc/c++%20字符类型总结区别wchar_t,char,WCHAR.docx)

### Visual Studio 2013已经不支持非Unicode的工程的编译



Character Set选择 Use Multi-Byte Character Set时，编译提示出错



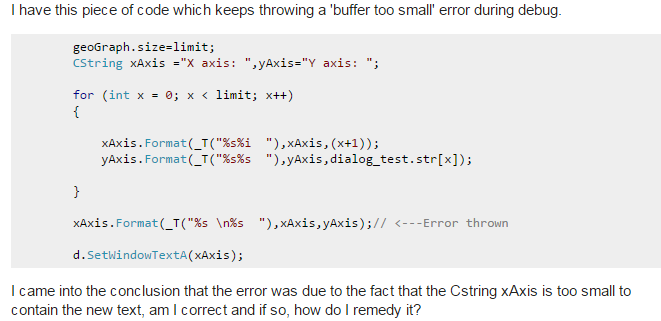
这也意味着今后的MFC工程，字符串最好定义成L开头的unicode或者用\_T(“fasdfad”) 之类的进行封装

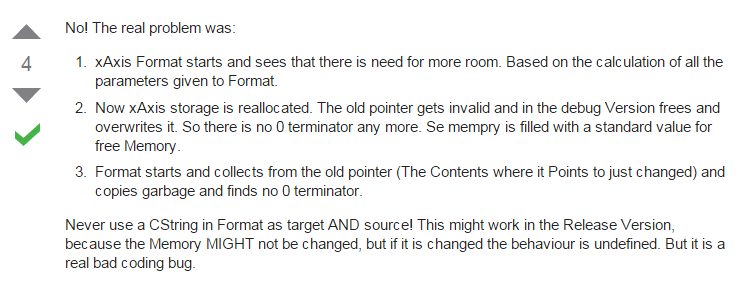
### CString Format: 源和目标不要使用同一个变量

Never use a CString in Format as target AND source! 原因是当遇到format函数时，目标变量要被初始化，因为可能要重新分配内存，字符串结尾的0字符可能就没有了。这样遇到源变量的时候，就可能要出错了。

现在只是debug版的时候总是出错。提示Buffer too small







# MFC结构

## CDialog

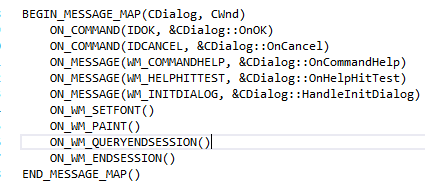
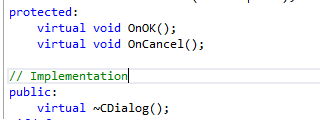
### 头文件和实现文件

D:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\atlmfc\src\mfc\dlgcore.cpp

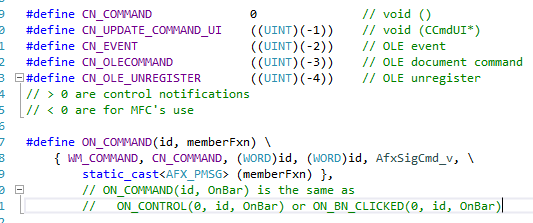
d:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\atlmfc\include\afxwin.h

### 类里声明了OnOK和OnCancel方法，和自己的MsgMap

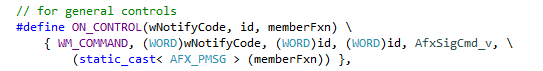
注意这里的方法虽然也是时间处理函数，但是并没有加任何的afx\_msg 前缀。



ON\_COMMAND 定义在d:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\atlmfc\include\afxmsg\_.h



可以比较同ON\_CONTROL的区别



 ， 这个id指的是控件的ID，比如



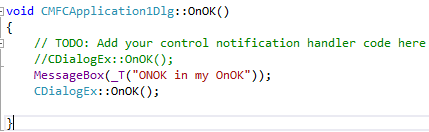
### 在自己的类里处理Dialog的IDOK按钮的click事件

首先，因为在CDialog里已经有IDOK的click时间的处理，所以自己的Dialog里可以不需要任何定义。如果想自定义

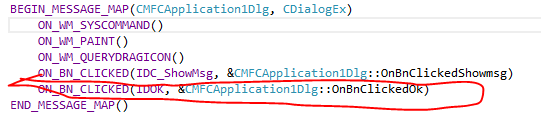
方法一： 重新定义虚函数OnOK就可以了

这样需要把OnOK的定义和实现都在自己的子类中加一遍。

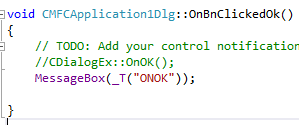
 按照C++ virtual 函数继承原则，继承的virtual函数可以加或者不加virtual关键字，但都会自动成为virtual函数



方法二： 用最直接的方法，自己捕获IDOK的client时间



这样父类的OnOK方法就不会被调用了，父类注册的MESSAGE\_MAP也不起作用了。



消息会先检索到子类然后向父类检索，因此子类可以屏蔽一些消息。屏蔽的原理如下：



其实就是先遍历本类自己的MessageEntries，如果没找到再遍历父类的。

## 消息机制

<http://blog.csdn.net/stan1989/article/details/8718899>

# Project

## ListCtrlDemo

[CListCtrl](ListCtrlDemo/doc/CListCtrl.docx)