MFC的消息处理机制

Windows应用程序依靠外部发生的事件来驱动，程序一直等待任何可能的输入，做出判断再进行处理。所有的GUI系统都是以消息为基础、以事件为驱动。

每一个windows应用程序都应该有一个如下回路：

while (GetMessage(&msg, ……)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

其中，msg是一种windows内定的一种格式：

typedef struct tagMSG{

HWND hwnd;

UINT message; // WM\_xxx，例如WM\_MOUSEMOVE， WM\_SIZE...

WPARAM wParam;

LPARAM lParam;

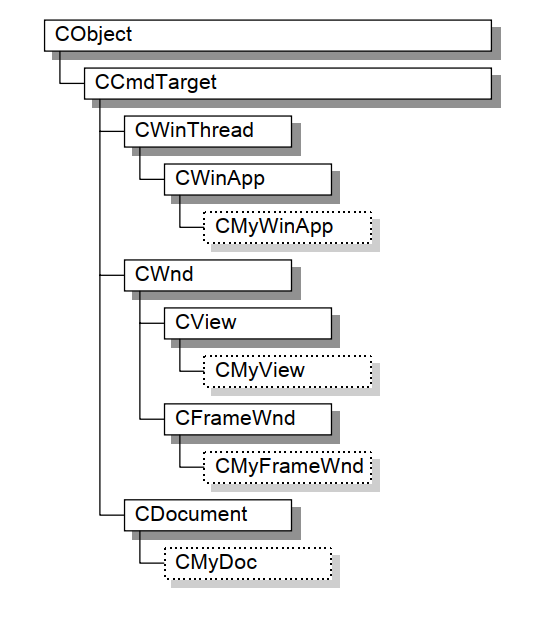
DWORD time;

POINT pt;

} MSG;

由窗口接受并处理消息。每一个窗口都有一个负责处理消息的函数，程序员

必须负责设计这个所谓的「窗口函数」（ window procedure，或称为window function）。如果窗口获得一个消息，这个窗口函数必须判断消息的类别，决定处理的方式（消息映射）。

依靠宏实现的消息映射：

在类声明中：

class CScribbleDoc : public CDocument

{

...

**DECLARE\_MESSAGE\_MAP()**

};

在类源码中：

**BEGIN\_MESSAGE\_MAP(**CScribbleDoc, CDocument**)**

//{{AFX\_MSG\_MAP(CScribbleDoc)

**ON\_COMMAND**(ID\_EDIT\_CLEAR\_ALL, OnEditClearAll)

**ON\_COMMAND**(ID\_PEN\_THICK\_OR\_THIN, OnPenThickOrThin)

...

//}}AFX\_MSG\_MAP

**END\_MESSAGE\_MAP()**

**#define DECLARE\_MESSAGE\_MAP()** \

private: \

static const **AFX\_MSGMAP\_ENTRY** \_messageEntries[ ]; \

protected: \

static AFX\_DATA const **AFX\_MSGMAP** messageMap; \

virtual const **AFX\_MSGMAP**\* GetMessageMap() const; \

struct **AFX\_MSGMAP\_ENTRY**

{

UINT nMessage; // windows message

UINT nCode;

UINT nID;

UINT nLastID;

UINT nSig;

**AFX\_PMSG** pfn; // routine to call (or special value)

};

其中，**AFX\_PMSG**是函数指针类型

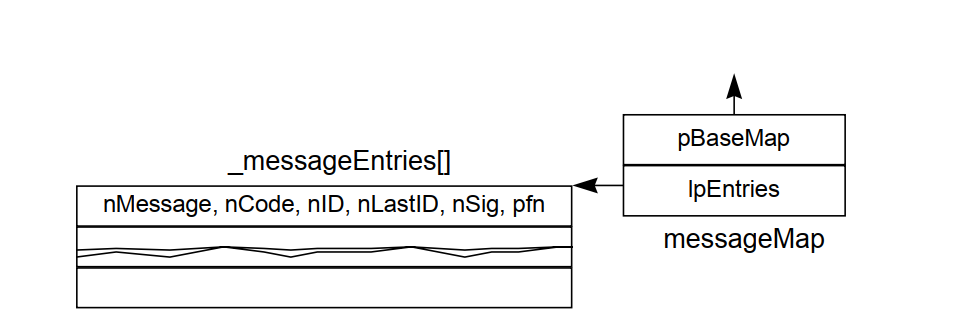
struct **AFX\_MSGMAP**

{

const **AFX\_MSGMAP**\* pBaseMap;

const **AFX\_MSGMAP\_ENTRY**\* lpEntries;

};



**#define BEGIN\_MESSAGE\_MAP**(theClass, baseClass) \

const **AFX\_MSGMAP**\* theClass::GetMessageMap() const \

{ return &theClass::messageMap; } \

AFX\_DATADEF const **AFX\_MSGMAP** theClass::messageMap = \

{ &baseClass::messageMap, &theClass::\_messageEntries[0] }; \

const **AFX\_MSGMAP\_ENTRY** theClass::\_messageEntries[] = \

{ \

**#define** **END\_MESSAGE\_MAP**() \

{0, 0, 0, 0, **AfxSig\_end**, (**AFX\_PMSG**)0 } \

}; \

**AfxSig\_end** 在AFXMSG\_.H 中被定义为0

**#define ON\_COMMAND**(id, memberFxn) \

{ WM\_COMMAND, CN\_COMMAND, (WORD)id, (WORD)id, AfxSig\_vv, (AFX\_PMSG)memberFxn },

**#define ON\_WM\_CREATE**() \

{ WM\_CREATE, 0, 0, 0, AfxSig\_is, \

(AFX\_PMSG)(AFX\_PMSGW)(int (AFX\_MSG\_CALL CWnd::\*)(LPCREATESTRUCT))OnCreate },

**#define ON\_WM\_DESTROY**() \

{ WM\_DESTROY, 0, 0, 0, AfxSig\_vv, \

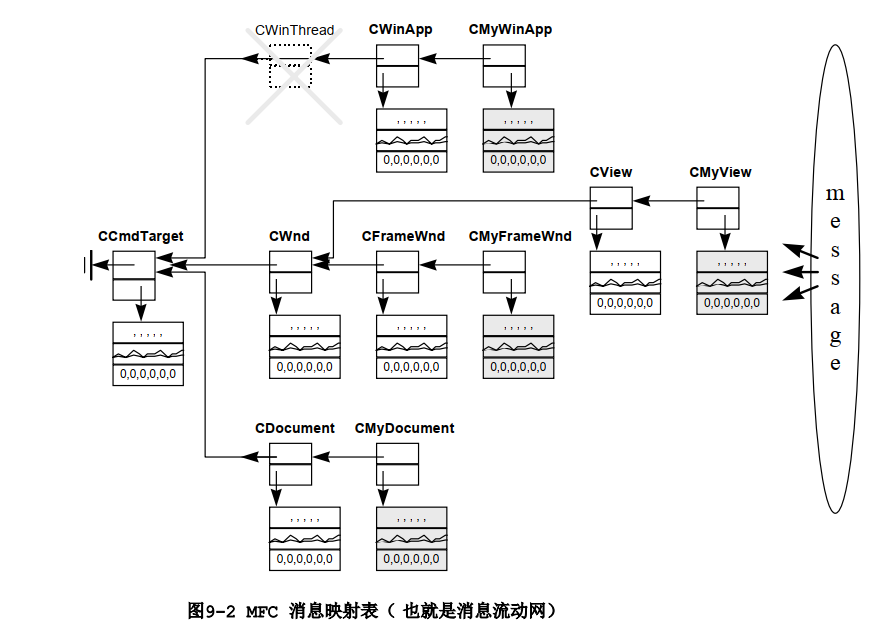
(AFX\_PMSG)(AFX\_PMSGW)(void (AFX\_MSG\_CALL CWnd::\*)(void))OnDestroy },

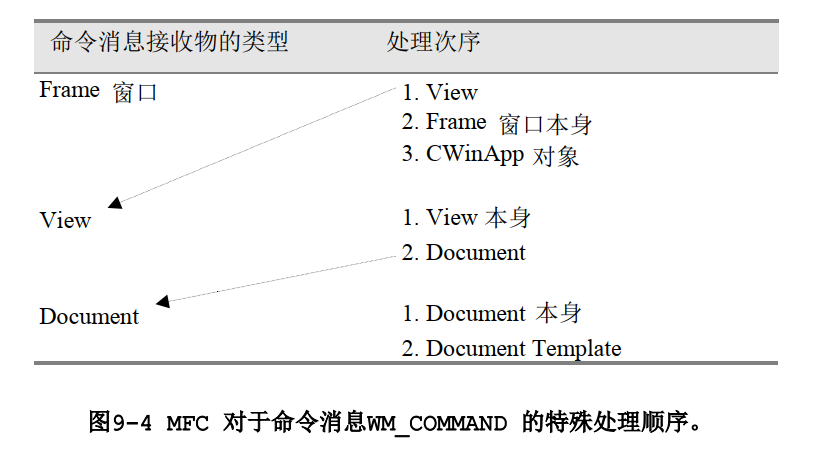
**#define ON\_WM\_MOVE**() \

{ WM\_MOVE, 0, 0, 0, AfxSig\_vvii, \

(AFX\_PMSG)(AFX\_PMSGW)(void (AFX\_MSG\_CALL CWnd::\*)(int, int))OnMove },

消息流动



** 对于命令消息WM\_COMMAND的特殊处理顺序：**