消息循环

1. 标准消息。任何派生自cwnd的类都可以接受该消息，并按照继承关系接受。2命令消息，即 WM\_COMMAND, 任何派生自ccmdtarget的 类，兼可接受该消息。
2. 宏中为其类 申请了一个全局结构和获得该结构的函数，而在BEGIN 和 NED 之间添写刚才的全局结构，将消息和对应的处理函数联系起来，并通过 basemap指针，将各类按照继承顺序连接起来，从而提供消息流动的道路。
3. 举例如下：

CMywnd:public Cwnd

{

DECLARE\_MESSAGE\_MAP();

}

BEGIN\_MESSAGE\_MAP(cMyWnd,CWnd)

ON\_WN\_CREATE()

ON\_WM\_PAINT()

END\_MESSAGE\_MAP()

被展开后，代码如下：

CMyWnd: public CWnd

{

Priate:

Static const AFX\_MSGMAP\_ENTRY \_messageEnteries[];

Protected:

Static AFX\_DATA const AFX\_MSGMAP messageMap;

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

}

Const AFX\_MSGMAP\* CMyWnd::GetMessageMap()const

{

Return & CMyWnd::messgaMap;

}

AFX\_DATA const AFX\_MGMAP CMyWnd::messageMap=

{ & CWnd::messgeMap,&CMyWnd::\_messageEntries[0]};

Const AFX\_MSGMAP\_ENTRY CMyWnd::\_messageEntries[]=

{

};

这样WM\_CREATE 等在消息网中流动，当流到CMyWnd类的messageMap结构时，发现有该消息的记录，则调用记录中记载的oncreate和onpaint函数，进行响应消息，从而完成了消息驱动机制。

系统都在不断的用GetMessage从消息队列中取出消息，然后用dispatchMessage将消息发送到窗口函数中去。