 对于回调函数的编写始终是写特殊处理功能程序时用到的技巧之一。先介绍一下回调的使用基本方法与原理。 1、在这里设：回调函数为A()(这是最简单的情况，不带参数，但我们应用的实际情况常常很会复杂)，使用回调函数的操作函数为B()， 但B函数是需要参数的，这个参数就是指向函数A的地址变量，这个变量一般就是函数指针。使用方法为：int A(char *p). // 回调函数 typedef int(*CallBack)(char *p) . // 声明CallBack 类型的函数指针 CallBack myCallBack . // 声明函数指针变量myCallBack = A. // 得到了函数A的地址 B函数一般会写为 B(CallBack lpCall,char * P,.....). // 此处省略了p后的参数形式 。 所以回调机制可解为，函数B要完成一定功能，但他自己是无法实现全部功能的。 需要借助于函数A来完成，也就是回调函数。B的实现为：B(CallBack lpCall,char *pProvide){ ..... // B 的自己实现功能语句 lpCall(PpProvide). // 借助回调完成的功能 ，也就是A函数来处理的。 ..... // B 的自己实现功能语句}// ---------- 使用例子 ----------char *p = "hello!".CallBack myCallBack . myCallBack = A .B(A, p). 以上就是回调的基本应用，本文所说的变身，其实是利用传入不同的函数地址，实现调用者类与回调函数所在类的不同转换 。 1、问题描述 CUploadFile 类完成数据上传，与相应的界面进度显示。 主要函数Send(...) 和回调函数 GetCurState() . class CUploadFile : public CDialog{ ..... int Send(LPCTSTR lpServerIP, LPCTSTR lpServerPort, LPCTSTR UploadFilePath) . static int

GetCurState(int nCurDone, int nInAll, void * pParam) . ..... }int CUploadFile ::Send(LPCTSTR lpServerIP, LPCTSTR lpServerPort, LPCTSTR UploadFilePath){ ... // 导出传输数据的函数 int ret = Upload((LPSTR)(LPCTSTR)m_strData, GetCurState, // 在这个回调函数中处理界面 this, // CUploadFile 的自身指针， 也就是pParam 所接受的参数 (LPSTR)(LPCTSTR)UploadFilePath, "", "",).}int CUploadFile ::GetCurState(int nCurData, int nInAll, void * pParam) { ..... UploadFile *pThis = (UploadFile *)pParam. // nCurData 当前以传出的数据量 // nInAll 总的数据量 // 有了pThis可以对界面进行各种操作了。 .....} 但大家仔细观察就可以发现，这个类把数据传送和界面显示聚和到了一起，不容易得到复用。而且在复用过程中需要改动较多的地方。请大家记住现在的回调函数传入的类本身的静态成员函数。 现在我们把数据的传送和界面的显示分离。回调则要传入的是界面处理类的静态函数。 界面处理类 CShowGUI,数据上传类 CUploadData class CUploadData { ..... typedef int(*SetUploadCaller)(int nCurData, int nInAll, void * pParam). int UploadFile(LPCTSTR lpFileNamePath,LPVOID lparam,SetUploadCaller Caller). // 接受外界出入的参数,主要是回调函数的地址通过参数Caller, int Send(LPCTSTR lpServerIP, LPCTSTR lpServerPort, LPCTSTR UploadFilePath) . ..... // 注意此时不在需要GetCurState 函数了 。}class CShowGUI: public CDialog{ ..... typedef int(*SetUploadCaller)(int nCurData, int nInAll, void * pParam). void SetCallBack(LPCTSTR strPath). static int GetCurState(int nCurData, int nInAll, void * pParam) . CUploadData m_Uploa d . // 数据上传类是界面显示类的一个

成员变量。 .....}void CShowGUI :: SetCallBack(LPCTSTR strPath){ CUploadData myUploadData . SetUploadCaller myCaller. // 声明一个函数指针变量 myCaller = CurState . // 取得界面处理函数的地址 myUploadData .UploadFile(strPath,this,myCaller). // 界面处理类的函数传入,实现了数据传入与界面处理的分离 .}通过上面的演示做到了界面与数据的分离,回调函数分别扮演了不同角色,所以随着处理问题的不同应灵活应用，但同样因为处理数据类不知道界面处理类或外部调用类的类型，而更无法灵活地处理界面的不同显示方式。这方面还希望喜欢钻研技术的朋友继续研究 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com