

利用Thunk让C 成员函数变回调函数计算机等级考试 PDF转换
可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/645/2021_2022__E5_88_A9_E7_94_A8Thun_c97_645545.htm Windows API经常需要回调函数，而在C 开发中面向对象当行其道，若能让C 类的成员函数成为回调函数，简直就是大善！但是C 成员函数都隐含了一个this指针用于指向当前的对象。要实现回调确实不容易。我大约一年前就接触到Thunk技术，甚至也看过利用Thunk实现将成员函数变成回调函数的例子。但是我实在没了解过C 汇编后的样子，很容易钻了牛角尖，看都看不懂，直接用他们的程序又不敢，毕竟出错后不好处理。前端时间偶尔想起Thunk技术，对未懂技术老这样悬着很可能影响自己的程序员生涯的，于是决心闭关参悟（没办法，资质差啊），终于弄明白了。那种感觉啊，就像诚信礼佛的人突然见到如来一样，或者换了贴近自己的比喻：就像千年色鬼见到美女一样的兴奋。我忍不住的模仿小说中的修真人士突悟大道后的感叹：原来如此！下面的分享一下我的收获，基本上是出入门径的写给初学者的，大侠千万要止步，小弟皮薄！稍微研究了一下C 汇编后的代码，一般调用C 的成员函数之前，都是使用ECX寄存器保存对象的指针，好在C 成员函数的调用约定__thiscall的参数压栈顺序和堆栈平衡的维护都是和回调函数的调用约定__stdcall一样，所以只需要构造汇编将对象指针保存在ECX寄存器后JMP到成员函数的执行地址就可以了。先写个C 结构拼凑这两条汇编码：

```
#pragma pack( push, 1 )
struct MemFunToStdCallThunk { BYTE m_mov. DWORD m_this.
BYTE m_jump. DWORD m_relproc. BOOL Init( DWORD_PTR
```

```
proc, void* pThis ) { m_mov = 0xB9. m_this = PtrToUlong(pThis).
m_jump = 0xe9. m_relproc = DWORD((INT_PTR)proc -
((INT_PTR)this sizeof(MemFunToStdCallThunk))).
::FlushInstructionCache( ::GetCurrentProcess(), this,
sizeof(MemFunToStdCallThunk) ). return TRUE. } void*
GetCodeAddress() { return this. }. #pragma pack( pop ) 这个结构
相当于两条汇编语句： mov ecx , pThis jmp [偏移地址] 使用
： class CTestClass { private: int m_nBase.
MemFunToStdCallThunk m_thunk. void memFun( int m, int n ) {
int nSun = m_nBase m n. CString str. str.Format( _T("%d"), nSun ).
AtlMessageBox( NULL, _U_STRINGOrID( str ) ). } public:
CTestClass() { m_nBase = 10. } void Test() { //UnionCastType:利
用联合将函数指针转换成DWORD_PTR m_thunk.Init(
UnionCastType 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
```