链滴

reinterpret_cast 和 static_cast

作者：localvar
原文链接：https://ld246.com/article/1521502296450
来源网站：链滴

最近写一个使用完成端口的应用时, 居然在最简单的类型转换上栽了一个跟头, 写出来与大家分享，以免犯和我类似的错误。为了能尽量统一的处理每一个io操作, 我定义了下面这个类:

```
class CIoPacket : public OVERLAPPED
{
public:
    virtual void OnIoComplete( ULONG_PTR key, DWORD dwBytes ) = 0;
    virtual void OnIoFailure( ULONG_PTR key, DWORD dwBytes ) = 0;
};
```

其中的两个虚函数分别在io操作成功完成或失败后被调用. 它的派生类将用于记录每一次io的相关信息. 为了从网络上接收数据, 我有从它派生出了一个类:

```
class CNetPkt : public CIoPacket
{
protected:
    // 定义一些成员变量
public:
    virtual void OnIoComplete( ULONG_PTR key, DWORD dwBytes )
    {
        // 执行一些操作
    }
    virtual void OnIoFailure( ULONG_PTR key, DWORD dwBytes )
    {
        // 执行一些操作
    }
};
```

下面的代码启动了一个接受数据的操作:

```
//...
SOCKET sck;
// 初始化sck并绑定到一个完成端口
CNetPkt* pPkt = new CNetPkt();
// 设置pPkt的相关字段
::WSARecv( sck, &wsabuf, 1, &dwBytes, &dwFlags, pPkt, NULL );              // (0)
//...
```

下面是完成端口线程中的代码:

```
DWORD dwBytes = 0;
ULONG_PTR key = 0;
LPOVERLAPPED pol = NULL;

if( ::GetQueuedCompletionStatus(g_hIocp, &dwBytes, &key, &pol, INFINITE) )
{
    if( key != 0 )
        reinterpret_cast(pol)->OnIoComplete( key, dwBytes );   // (1)
}
else
{
    if( pol != NULL )
        reinterpret_cast(pol)->OnIoFailure( key, dwBytes );    // (2)
}
```

自我感觉实现的既灵活又漂亮还健壮、高效. 但是程序每次运行到(1)或(2)时就会出现非法操作, 令我思不得其解. 首先仔细检查程序, 没发现错误; 又使出十八般调试功夫, 还是没有找到问题所在. 正在头之时, 突然发现(1)处的pol和(0)处的pPkt的值并不一样, pol比pPkt大了4, 进一步通过反汇编发现(0)处际传给WSARecv的就是((LPBYTE)pPkt)+4. 两个值不一样, 总出错也就不奇怪了.

可编译器为什么要给指针加上4呢? 难道是编译器把CIoPacket的vfptr放到了OVERLAPPED的前面? 是我记得vc应该是把它放在后面的呀(具体不敢确定了, 但好像VC6是放在后面). 一番测试证实了我的测, vc7.1就是会把vfptr放到类结构的最前面, 该死的微软居然偷偷改了这么重要的编译细节. 但也不能骂微软, 自己的错误也要检讨一下, 上面的程序中我应该用static_cast, 而不是reinterpret_cast, 因为static_cast能正确调整基类和派生类的指针, 而reinterpret_cast从汇编的角度看是什么也不干的. 如果vfpt放在后面, static_cast和reinterpret_cast的结果是一样的, 但当vfptr放在前面的时候它们就不同了. 由C++标准没有规定vfptr的放置位置, 所以大家进行类型转换时一定要注意选择正确方式, 避免出现我样的、隐蔽的可移植性问题.