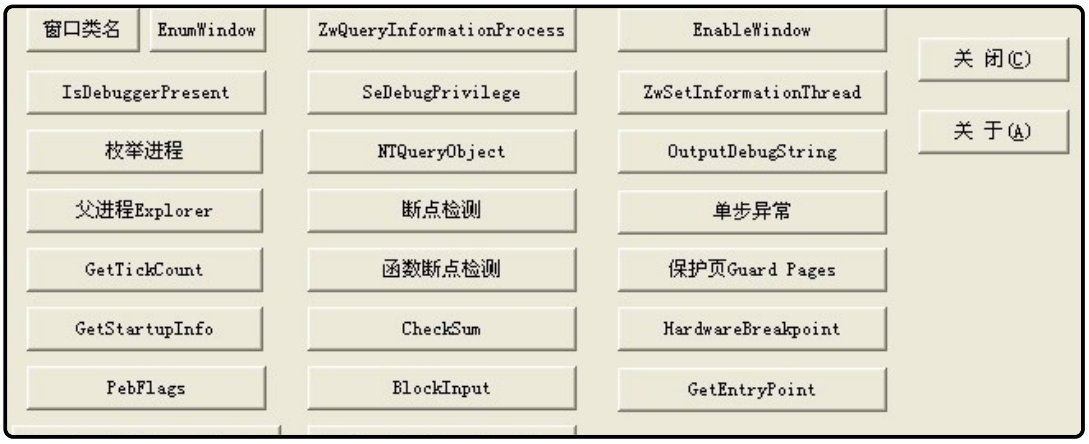


```cpp
// DetectODDlg.cpp : implementation file
//

#include "stdafx.h"
#include "DetectOD.h"
#include "DetectODDlg.h"
#include "Shlwapi.h"
#include "tlhelp32.h"
#include "Windows.h"
#include "Winable.h"
#include "eh.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
static DWORD NewEip;
/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    //{{AFX_MSG(CAboutDlg)
    afx_msg void OnMypage();
    afx_msg void OnMouseMove(UINT nFlags, CPoint point);
    virtual BOOL OnInitDialog();
    afx_msg void OnComeon();
    afx_msg void OnMyicon();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //{{AFX_DATA_INIT(CAboutDlg)
    //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
```

```
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAboutDlg)
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    //{{AFX_MSG_MAP(CAboutDlg)
    ON_BN_CLICKED(IDC_MYPAGE, OnMypage)
    ON_WM_MOUSEMOVE()
    ON_BN_CLICKED(IDC_COMEON, OnComeon)
    ON_BN_CLICKED(IDC_MYICON, OnMyicon)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CDetectODDlg dialog

CDetectODDlg::CDetectODDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CDetectODDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CDetectODDlg)
        // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CDetectODDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CDetectODDlg)
        // NOTE: the ClassWizard will add DDX and DDV calls here
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CDetectODDlg, CDialog)
    //{{AFX_MSG_MAP(CDetectODDlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_WNDCLS, OnWndcls)
    ON_BN_CLICKED(IDC_ISDEBUGGERPRESENT, OnIsdebuggerpresent)
    ON_BN_CLICKED(IDC_ENUMWINDOW, OnEnumwindow)
    ON_BN_CLICKED(IDC_EnumProcess, OnEnumProcess)
    ON_BN_CLICKED(IDC_Explorer, OnExplorer)
    ON_BN_CLICKED(IDC_GetTickCount, OnGetTickCount)
    ON_BN_CLICKED(IDC_GetStartupInfo, OnGetStartupInfo)
    ON_BN_CLICKED(IDC_PEBFLAGS, OnPebflags)
    ON_BN_CLICKED(IDC_CHECKREMOTEDEBUGGERPRESENT, OnCheckremotedebuggerpresent)
    ON_BN_CLICKED(IDC_SetUnhandledExceptionFilter, OnSetUnhandledExceptionFilter)
    ON_BN_CLICKED(IDC_ZwQueryInformationProcess, OnZwQueryInformationProcess)
    ON_BN_CLICKED(IDC_SeDebugPrivilege, OnSeDebugPrivilege)
    ON_BN_CLICKED(IDC_NTQueryObject, OnNTQueryObject)
    ON_BN_CLICKED(IDC_DectectBreakpoints, OnDectectBreakpoints)
    ON_BN_CLICKED(IDC_DectectFuncBreakpoints, OnDectectFuncBreakpoints)
    ON_BN_CLICKED(IDC_BlockInput, OnBlockInput)
    ON_BN_CLICKED(IDC_CHECKSUM, OnChecksum)
    ON_BN_CLICKED(IDC_EnableWindow, OnEnableWindow)
    ON_BN_CLICKED(IDC_ZwSetInformationThread, OnZwSetInformationThread)
    ON_BN_CLICKED(IDC_OutputDebugString, OnOutputDebugString)
    ON_BN_CLICKED(IDC_GetEntryPoint, OnGetEntryPoint)
    ON_BN_CLICKED(IDC_TrapFlag, OnTrapFlag)
    ON_BN_CLICKED(IDC_GuardPages, OnGuardPages)
    ON_BN_CLICKED(IDC_HARDWAREBREAKPOINT, OnHardwarebreakpoint)
    ON_BN_CLICKED(IDC_ABOUT, OnAbout)
    ON_BN_CLICKED(IDC_MYPAGE2, OnMypage2)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CDetectODDlg message handlers

BOOL CDetectODDlg::OnInitDialog()
```

```
{
    CDialog::OnInitDialog();

    // Add "About..." menu item to system menu.

    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }

    // Set the icon for this dialog.  The framework does this automatically
    //  when the application's main window is not a dialog
//  SetIcon(m_hIcon, TRUE);         // Set big icon
//  SetIcon(m_hIcon, FALSE);        // Set small icon

    // TODO: Add extra initialization here
    SetClassLong(m_hWnd,GCL_HICON,(LONG)(LoadIcon(AfxGetApp()-
>m_hInstance,MAKEINTRESOURCE(IDI_DOG))));
    return TRUE;  // return TRUE  unless you set the focus to a control
}

void CDetectODDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CDetectODDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

```

```
// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CDetectODDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

void CDetectODDlg::OnWndcls()
{
    // TODO: Add your control notification handler code here
    HWND hWnd;
    if(hWnd=::FindWindow("OllyDbg",NULL))
    {
        MessageBox("发现OD");
    }else{
        MessageBox("没发现OD");
    }

}
void CDetectODDlg::OnIsdebuggerpresent()
{
    // TODO: Add your control notification handler code here
    if(IsDebuggerPresent())
    {
        MessageBox("发现OD");
    }
    else
    {
        MessageBox("没有OD");
    }
}
/************************************************/
BOOL CALLBACK EnumWindowsProc(
  HWND hwnd,      // handle to parent window
  LPARAM lParam   // application-defined value
  )
{
    char ch[100];
    CString str="Ollydbg";
    if(IsWindowVisible(hwnd))
    {
        ::GetWindowText(hwnd,ch,100);
        //AfxMessageBox(ch);
        if(::StrStrI(ch,str))
        {
            AfxMessageBox("发现OD");
            return FALSE;
        }
    }
    return TRUE;
}

void CDetectODDlg::OnEnumwindow()
{
    // TODO: Add your control notification handler code here
    EnumWindows(EnumWindowsProc,NULL);
    AfxMessageBox("枚举窗口结束，未提示发现OD，则没有OD");
}

/************************************************/
void CDetectODDlg::OnEnumProcess()
{
    // TODO: Add your control notification handler code here

    HANDLE hwnd;
    PROCESSENTRY32 tp32;  //结构体
    CString str="OLLYDBG.EXE";
    BOOL bFindOD=FALSE;
    hwnd=::CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,NULL);
    if(INVALID_HANDLE_VALUE!=hwnd)
    {
        Process32First(hwnd,&tp32);
        do{
            if(0==lstrcmpi(str,tp32.szExeFile))
```

```
            {
                AfxMessageBox("发现OD");
                bFindOD=TRUE;
                break;
            }
        }while(Process32Next(hwnd,&tp32));
        if(!bFindOD)
            AfxMessageBox("没有OD");
    }
    CloseHandle(hwnd);
}

void CDetectODDlg::OnExplorer()
{
    // TODO: Add your control notification handler code here
    HANDLE hwnd;
    PROCESSENTRY32 tp32;  //结构体
    CString str="Explorer.EXE";

    DWORD ExplorerID;
    DWORD SelfID;
    DWORD SelfParentID;
    SelfID=GetCurrentProcessId();
    ::GetWindowThreadProcessId(::FindWindow("Progman",NULL),&ExplorerID);
    hwnd=::CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,NULL);
    if(INVALID_HANDLE_VALUE!=hwnd)
    {
        Process32First(hwnd,&tp32);
        do{
            if(0==lstrcmp(str,tp32.szExeFile))
            {
            //  ExplorerID=tp32.th32ProcessID;
            //  AfxMessageBox("aaa");
            }
            if(SelfID==tp32.th32ProcessID)
            {
                SelfParentID=tp32.th32ParentProcessID;
            }
        }while(Process32Next(hwnd,&tp32));

        str.Format("本进程: %d 父进程: %d Explorer进程: %d
",SelfID,SelfParentID,ExplorerID);
        MessageBox(str);
        if(ExplorerID==SelfParentID)
        {
            AfxMessageBox("没有OD");
        }
        else
        {
            AfxMessageBox("发现OD");
        }
    }
    CloseHandle(hwnd);
}

void CDetectODDlg::OnGetTickCount()
{
    // TODO: Add your control notification handler code here
    DWORD dTime1;
    DWORD dTime2;
    dTime1=GetTickCount();
    GetCurrentProcessId();
    GetCurrentProcessId();
    GetCurrentProcessId();
    GetCurrentProcessId();
    dTime2=GetTickCount();
    if(dTime2-dTime1>100)
    {
        AfxMessageBox("发现OD");
    }
    else{
        AfxMessageBox("没有OD");
    }
}
```

```
350
351 void CDetectODDlg::OnGetStartupInfo()
352 {
353     // TODO: Add your control notification handler code here
354     STARTUPINFO info={0};
355     GetStartupInfo(&info);
356     if(info.dwX!=0 || info.dwY!=0 || info.dwXCountChars!=0 || info.dwYCountChars!=0
357         || info.dwFillAttribute!=0 || info.dwXSize!=0 || info.dwYSize!=0)
358     {
359         AfxMessageBox("发现OD");
360     }
361     else{
362         AfxMessageBox("没有OD");
363     }
364
365 }
366
367 //********************************************
368 typedef ULONG NTSTATUS;
369 typedef ULONG PPEB;
370 typedef ULONG KAFFINITY;
371 typedef ULONG KPRIORITY;
372
373 typedef struct _PROCESS_BASIC_INFORMATION { // Information Class 0
374 NTSTATUS ExitStatus;
375 PPEB PebBaseAddress;
376 KAFFINITY AffinityMask;
377 KPRIORITY BasePriority;
378 ULONG UniqueProcessId;
379 ULONG InheritedFromUniqueProcessId;
380 } PROCESS_BASIC_INFORMATION, *PPROCESS_BASIC_INFORMATION;
381
382 typedef enum _PROCESSINFOCLASS {
383 ProcessBasicInformation, // 0 Y N
384 ProcessQuotaLimits, // 1 Y Y
385 ProcessIoCounters, // 2 Y N
386 ProcessVmCounters, // 3 Y N
387 ProcessTimes, // 4 Y N
388 ProcessBasePriority, // 5 N Y
389 ProcessRaisePriority, // 6 N Y
390 ProcessDebugPort, // 7 Y Y
391 ProcessExceptionPort, // 8 N Y
392 ProcessAccessToken, // 9 N Y
393 ProcessLdtInformation, // 10 Y Y
394 ProcessLdtSize, // 11 N Y
395 ProcessDefaultHardErrorMode, // 12 Y Y
396 ProcessIoPortHandlers, // 13 N Y
397 ProcessPooledUsageAndLimits, // 14 Y N
398 ProcessWorkingSetWatch, // 15 Y Y
399 ProcessUserModeIOPL, // 16 N Y
400 ProcessEnableAlignmentFaultFixup, // 17 N Y
401 ProcessPriorityClass, // 18 N Y
402 ProcessWx86Information, // 19 Y N
403 ProcessHandleCount, // 20 Y N
404 ProcessAffinityMask, // 21 N Y
405 ProcessPriorityBoost, // 22 Y Y
406 ProcessDeviceMap,// 23 Y Y
407 ProcessSessionInformation, // 24 Y Y
408 ProcessForegroundInformation, // 25 N Y
409 ProcessWow64Information // 26 Y N
410 } PROCESSINFOCLASS;
411
412 typedef NTSTATUS (_stdcall *ZwQueryInformationProcess)(
413 HANDLE ProcessHandle,
414 PROCESSINFOCLASS ProcessInformationClass,
415 PVOID ProcessInformation,
416 ULONG ProcessInformationLength,
417 PULONG ReturnLength
418 ); //定义函数指针
419 void CDetectODDlg::OnPebflags()
420 {
421     // TODO: Add your control notification handler code here
422
423     //定义函数指针变量
```

```
    ZwQueryInformationProcess MyZwQueryInformationProcess;

    HANDLE hProcess = NULL;
    PROCESS_BASIC_INFORMATION pbi = {0};
    ULONG peb = 0;
    ULONG cnt = 0;
    ULONG PebBase = 0;
    ULONG AddrBase;
    BOOL bFoundOD=FALSE;
    WORD flag;
    DWORD dwFlag;
    DWORD bytesrw;
    DWORD ProcessId=GetCurrentProcessId();
    hProcess = OpenProcess(PROCESS_QUERY_INFORMATION | PROCESS_VM_READ, FALSE,
ProcessId);
    if (hProcess != NULL) {
        //函数指针变量赋值
        MyZwQueryInformationProcess=
(ZwQueryInformationProcess)GetProcAddress(LoadLibrary("ntdll.dll"),"ZwQueryInformation
Process");
        //函数指针变量调用
        if (MyZwQueryInformationProcess(
                hProcess,
                ProcessBasicInformation,
                &pbi,
                sizeof(PROCESS_BASIC_INFORMATION),
                &cnt) == 0)
        {
            PebBase = (ULONG)pbi.PebBaseAddress;
            AddrBase=PebBase;
            if (ReadProcessMemory(hProcess,(LPCVOID)(PebBase+0x68),&flag,2,&bytesrw)
&& bytesrw==2)
            { //PEB.NtGlobalFlag
                if(0x70==flag){
                    bFoundOD=TRUE;
                }
            }
            if (ReadProcessMemory(hProcess,(LPCVOID)(PebBase+0x18),&dwFlag,4,&bytesrw)
&& bytesrw==4)
            {
                AddrBase=dwFlag;
            }
            if (ReadProcessMemory(hProcess,(LPCVOID)(AddrBase+0x0c),&flag,2,&bytesrw)
&& bytesrw==2)
            {//PEB.ProcessHeap.Flags
                if(2!=flag){
                    bFoundOD=TRUE;
                }
            }
            if (ReadProcessMemory(hProcess,(LPCVOID)(AddrBase+0x10),&flag,2,&bytesrw)
&& bytesrw==2)
            {//PEB.ProcessHeap.ForceFlags
                if(0!=flag){
                    bFoundOD=TRUE;
                }
            }
            if(bFoundOD==FALSE)
            {
                AfxMessageBox("没有OD");
            }
            else
            {
                AfxMessageBox("发现OD");
            }
        }
        CloseHandle(hProcess);
    }
}

//************************************************************
typedef BOOL (WINAPI *CHECK_REMOTE_DEBUGGER_PRESENT)(HANDLE, PBOOL);

void CDetectODDlg::OnCheckremotedebuggerpresent()
{
```

```
    // TODO: Add your control notification handler code here
    HANDLE      hProcess;
    HINSTANCE   hModule;
    BOOL        bDebuggerPresent = FALSE;
    CHECK_REMOTE_DEBUGGER_PRESENT CheckRemoteDebuggerPresent;
    hModule = GetModuleHandleA("Kernel32");
    CheckRemoteDebuggerPresent =
        (CHECK_REMOTE_DEBUGGER_PRESENT)GetProcAddress(hModule,
"CheckRemoteDebuggerPresent");
    hProcess = GetCurrentProcess();
    CheckRemoteDebuggerPresent(hProcess,&bDebuggerPresent);
    if(bDebuggerPresent==TRUE)
    {
        AfxMessageBox("发现OD");
    }
    else
    {
        AfxMessageBox("没有OD");
    }
}
//*******************************************************
typedef NTSTATUS (_stdcall *ZW_QUERY_INFORMATION_PROCESS)(
HANDLE ProcessHandle,
PROCESSINFOCLASS ProcessInformationClass, //该参数也需要上面声明的数据结构
PVOID ProcessInformation,
ULONG ProcessInformationLength,
PULONG ReturnLength
); //定义函数指针

void CDetectODDlg::OnZwQueryInformationProcess()
{
    // TODO: Add your control notification handler code here
    HANDLE      hProcess;
    HINSTANCE   hModule;
    DWORD       dwResult;
    ZW_QUERY_INFORMATION_PROCESS MyFunc;
    hModule = GetModuleHandle("ntdll.dll");
    MyFunc=
(ZW_QUERY_INFORMATION_PROCESS)GetProcAddress(hModule,"ZwQueryInformationProcess");
    hProcess = GetCurrentProcess();
    MyFunc(
        hProcess,
        ProcessDebugPort,
        &dwResult,
        4,
        NULL);
    if(dwResult!=0)
    {
        AfxMessageBox("发现OD");
    }
    else
    {
        AfxMessageBox("没有OD");
    }
}
//*******************************************************
static DWORD lpOldHandler;
typedef LPTOP_LEVEL_EXCEPTION_FILTER (_stdcall  *pSetUnhandledExceptionFilter)(
                      LPTOP_LEVEL_EXCEPTION_FILTER lpTopLevelExceptionFilter
                      );
pSetUnhandledExceptionFilter lpSetUnhandledExceptionFilter;

LONG WINAPI TopUnhandledExceptionFilter(
    struct _EXCEPTION_POINTERS *ExceptionInfo
)
{
    _asm pushad
    AfxMessageBox("回调函数");
    lpSetUnhandledExceptionFilter((LPTOP_LEVEL_EXCEPTION_FILTER )lpOldHandler);
    ExceptionInfo->ContextRecord->Eip=NewEip;//转移到安全位置
    _asm popad
    return EXCEPTION_CONTINUE_EXECUTION;
}

```

```
void CDetectODDlg::OnSetUnhandledExceptionFilter()
{
    bool isDebugged=0;
    // TODO: Add your control notification handler code here
    lpSetUnhandledExceptionFilter =
(pSetUnhandledExceptionFilter)GetProcAddress(LoadLibrary(("kernel32.dll")),
  "SetUnhandledExceptionFilter");
    lpOldHandler=(DWORD)lpSetUnhandledExceptionFilter(TopUnhandledExceptionFilter);
    _asm{  //获取这个安全地址
        call me     //方式一，需要NewEip加上一个偏移值
me:
        pop NewEip  //方式一结束
        mov NewEip,offset safe //方式二，更简单
        int 3  //触发异常
    }
    AfxMessageBox("检测到OD");
    isDebugged=1;
    _asm{
safe:
    }
    if(1==isDebugged){

    }else{
        AfxMessageBox("没有OD");
    }
}
//*****************************************************
void CDetectODDlg::OnSeDebugPrivilege()
{
    // TODO: Add your control notification handler code here
    HANDLE hProcessSnap;
    HANDLE hProcess;
    PROCESSENTRY32 tp32;  //结构体
    CString str="csrss.exe";
    hProcessSnap=::CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,NULL);
    if(INVALID_HANDLE_VALUE!=hProcessSnap)
    {
        Process32First(hProcessSnap,&tp32);
        do{
            if(0==lstrcmpi(str,tp32.szExeFile))
            {

hProcess=OpenProcess(PROCESS_QUERY_INFORMATION,NULL,tp32.th32ProcessID);
                if(NULL!=hProcess)
                {
                    AfxMessageBox("发现OD");
                }
                else
                {
                    AfxMessageBox("没有OD");
                }
                CloseHandle(hProcess);
            }
        }while(Process32Next(hProcessSnap,&tp32));
    }
    CloseHandle(hProcessSnap);
}

//*************************************************************
#ifndef STATUS_INFO_LENGTH_MISMATCH
#define STATUS_INFO_LENGTH_MISMATCH ((UINT32)0xC0000004L)
#endif

typedef enum _POOL_TYPE {
  NonPagedPool,
  PagedPool,
  NonPagedPoolMustSucceed,
  DontUseThisType,
  NonPagedPoolCacheAligned,
  PagedPoolCacheAligned,
  NonPagedPoolCacheAlignedMustS
} POOL_TYPE;

typedef struct _UNICODE_STRING {
```

```
    USHORT Length;
    USHORT MaximumLength;
    PWSTR Buffer;
} UNICODE_STRING;
typedef UNICODE_STRING *PUNICODE_STRING;
typedef const UNICODE_STRING *PCUNICODE_STRING;

typedef enum _OBJECT_INFORMATION_CLASS
{
    ObjectBasicInformation,         // Result is OBJECT_BASIC_INFORMATION structure
    ObjectNameInformation,          // Result is OBJECT_NAME_INFORMATION structure
    ObjectTypeInformation,          // Result is OBJECT_TYPE_INFORMATION structure
    ObjectAllTypesInformation,          // Result is OBJECT_ALL_INFORMATION structure
    ObjectDataInformation           // Result is OBJECT_DATA_INFORMATION structure

} OBJECT_INFORMATION_CLASS, *POBJECT_INFORMATION_CLASS;

typedef struct _OBJECT_TYPE_INFORMATION {
    UNICODE_STRING TypeName;
    ULONG TotalNumberOfHandles;
    ULONG TotalNumberOfObjects;
    WCHAR Unused1[8];
    ULONG HighWaterNumberOfHandles;
    ULONG HighWaterNumberOfObjects;
    WCHAR Unused2[8];
    ACCESS_MASK InvalidAttributes;
    GENERIC_MAPPING GenericMapping;
    ACCESS_MASK ValidAttributes;
    BOOLEAN SecurityRequired;
    BOOLEAN MaintainHandleCount;
    USHORT MaintainTypeList;
    POOL_TYPE PoolType;
    ULONG DefaultPagedPoolCharge;
    ULONG DefaultNonPagedPoolCharge;
} OBJECT_TYPE_INFORMATION, *POBJECT_TYPE_INFORMATION;

typedef struct _OBJECT_ALL_INFORMATION {
    ULONG NumberOfObjectsTypes;
    OBJECT_TYPE_INFORMATION ObjectTypeInformation[1];
} OBJECT_ALL_INFORMATION, *POBJECT_ALL_INFORMATION;

typedef struct _OBJECT_ALL_TYPES_INFORMATION {
    ULONG NumberOfTypes;
    OBJECT_TYPE_INFORMATION TypeInformation[1];
} OBJECT_ALL_TYPES_INFORMATION, *POBJECT_ALL_TYPES_INFORMATION;

typedef UINT32 (__stdcall  *ZwQueryObject_t) (
    IN HANDLE ObjectHandle,
    IN OBJECT_INFORMATION_CLASS ObjectInformationClass,
    OUT PVOID ObjectInformation,
    IN ULONG Length,
    OUT PULONG ResultLength );

void CDetectODDlg::OnNTQueryObject()
{
    // TODO: Add your control notification handler code here
    // 调试器必须正在调试才能检测到，仅打开OD是检测不到的
    HMODULE hNtDLL;
    DWORD dwSize;
    UINT i;
    UCHAR  KeyType=0;
    OBJECT_ALL_TYPES_INFORMATION *Types;
    OBJECT_TYPE_INFORMATION *t;
    ZwQueryObject_t ZwQueryObject;

    hNtDLL = GetModuleHandle("ntdll.dll");
    if(hNtDLL){
        ZwQueryObject = (ZwQueryObject_t)GetProcAddress(hNtDLL, "ZwQueryObject");
        UINT32 iResult = ZwQueryObject(NULL, ObjectAllTypesInformation, NULL, NULL,
&dwSize);
        if(iResult==STATUS_INFO_LENGTH_MISMATCH)
        {
            Types =
(OBJECT_ALL_TYPES_INFORMATION*)VirtualAlloc(NULL,dwSize,MEM_COMMIT,PAGE_READWRITE);
```

```
            if (Types == NULL)  return;
            if (iResult=ZwQueryObject(NULL,ObjectAllTypesInformation, Types, dwSize,
&dwSize)) return;
            for (t=Types->TypeInformation,i=0;i<Types->NumberOfTypes;i++)
            {
                if ( !_wcsicmp(t->TypeName.Buffer,L"DebugObject")) //比较两个是否相等,
这个L很特殊，本地的意思
                {
                    if(t->TotalNumberOfHandles > 0 || t->TotalNumberOfObjects > 0)
                    {
                        AfxMessageBox("发现OD");
                        VirtualFree (Types,0,MEM_RELEASE);
                        return;
                    }
                    break; // Found Anyways
                }
                t=(OBJECT_TYPE_INFORMATION *)((char *)t->TypeName.Buffer+((t-
>TypeName.MaximumLength+3)&~3));
            }
        }
        AfxMessageBox("没有OD!");
        VirtualFree (Types,0,MEM_RELEASE);
    }
}
/*******************************************************/
BOOL DetectBreakpoints()
{
    BOOL bFoundOD;
    bFoundOD=FALSE;
    __asm
    {
                jmp     CodeEnd
   CodeStart:   mov     eax,ecx  ;被保护的程序段
                nop
                push    eax
                push    ecx
                pop     ecx
                pop     eax
   CodeEnd:
                cld                 ;检测代码开始
                mov     edi,offset CodeStart
                mov     edx,offset CodeStart
                mov     ecx,offset CodeEnd
                sub     ecx,edx

                mov     al,0CCH
                repne   scasb
                jnz      ODNotFound
                mov bFoundOD,1
    ODNotFound:
    }
    return bFoundOD;
}
void CDetectODDlg::OnDectectBreakpoints()
{
    // TODO: Add your control notification handler code here
    if(DetectBreakpoints())
    {
        AfxMessageBox("发现OD");
    }
    else
    {
        AfxMessageBox("没有OD");
    }
}
/*******************************************************/
BOOL DetectFuncBreakpoints()
{
    BOOL bFoundOD;
    bFoundOD=FALSE;
    DWORD dwAddr;
    dwAddr=(DWORD)::GetProcAddress(LoadLibrary("user32.dll"),"MessageBoxA");
    __asm
    {
```

```
                cld                 ;检测代码开始
                mov     edi,dwAddr
                mov     ecx,100   ;100bytes
                mov     al,0CCH
                repne   scasb
                jnz     ODNotFound
                mov bFoundOD,1
    ODNotFound:
    }
    return bFoundOD;
}
void CDetectODDlg::OnDectectFuncBreakpoints()
{
    // TODO: Add your control notification handler code here
    if(DetectFuncBreakpoints())
    {
        AfxMessageBox("发现OD");
    }
    else
    {
        AfxMessageBox("没有OD");
    }
}

void CDetectODDlg::OnBlockInput()
{   // #include "Winable.h"
    // TODO: Add your control notification handler code here
    DWORD dwNoUse;
    DWORD dwNoUse2;
    ::BlockInput(TRUE);
    dwNoUse=2;
    dwNoUse2=3;
    dwNoUse=dwNoUse2;
    ::BlockInput(FALSE);
}
/*******************************************************/
BOOL CheckSum()
{
    BOOL bFoundOD;
    bFoundOD=FALSE;
    DWORD CHECK_SUM=5555; //正确校验值
    DWORD dwAddr;
    dwAddr=(DWORD)CheckSum;
    __asm
    {
                                ;检测代码开始
                mov     esi,dwAddr
                mov     ecx,100
                xor     eax,eax
 checksum_loop:
                movzx   ebx,byte ptr [esi]
                add     eax,ebx
                rol     eax,1
                inc     esi
                loop    checksum_loop

                cmp     eax,CHECK_SUM
                jz      ODNotFound
                mov     bFoundOD,1
    ODNotFound:
    }
    return bFoundOD;
}
void CDetectODDlg::OnChecksum()
{
    // TODO: Add your control notification handler code here
    if(CheckSum())
    {
        AfxMessageBox("发现OD");
    }
    else
    {
        AfxMessageBox("没有OD");
    }
```

```
}
/*******************************************************/

void CDetectODDlg::OnEnableWindow()
{
    // TODO: Add your control notification handler code here
    CWnd *wnd;
    wnd=GetForegroundWindow();
    wnd->EnableWindow(FALSE);
    DWORD dwNoUse;
    DWORD dwNoUse2;
    dwNoUse=2;
    dwNoUse2=3;
    dwNoUse=dwNoUse2;
    wnd->EnableWindow(TRUE);
}
/*******************************************************/
typedef enum _THREADINFOCLASS {
ThreadBasicInformation, // 0 Y N
ThreadTimes, // 1 Y N
ThreadPriority, // 2 N Y
ThreadBasePriority, // 3 N Y
ThreadAffinityMask, // 4 N Y
ThreadImpersonationToken, // 5 N Y
ThreadDescriptorTableEntry, // 6 Y N
ThreadEnableAlignmentFaultFixup, // 7 N Y
ThreadEventPair, // 8 N Y
ThreadQuerySetWin32StartAddress, // 9 Y Y
ThreadZeroTlsCell, // 10 N Y
ThreadPerformanceCount, // 11 Y N
ThreadAmILastThread, // 12 Y N
ThreadIdealProcessor, // 13 N Y
ThreadPriorityBoost, // 14 Y Y
ThreadSetTlsArrayAddress, // 15 N Y
ThreadIsIoPending, // 16 Y N
ThreadHideFromDebugger // 17 N Y
} THREAD_INFO_CLASS;

typedef NTSTATUS (NTAPI *ZwSetInformationThread)(
IN  HANDLE                      ThreadHandle,
IN  THREAD_INFO_CLASS           ThreadInformaitonClass,
IN  PVOID                       ThreadInformation,
IN  ULONG                       ThreadInformationLength
);

void CDetectODDlg::OnZwSetInformationThread()
{
    // TODO: Add your control notification handler code here
    CString str="利用我定位";
    HANDLE hwnd;
    HMODULE hModule;
    hwnd=GetCurrentThread();
    hModule=LoadLibrary("ntdll.dll");
    ZwSetInformationThread myFunc;
    myFunc=(ZwSetInformationThread)GetProcAddress(hModule,"ZwSetInformationThread");
    myFunc(hwnd,ThreadHideFromDebugger,NULL,NULL);
}
/*******************************************************/
void CDetectODDlg::OnOutputDebugString()
{
    // TODO: Add your control notification handler code here
    ::OutputDebugString("%s%s%s");
}
/*******************************************************/
void CDetectODDlg::OnGetEntryPoint()
{
    // TODO: Add your control notification handler code here
    IMAGE_DOS_HEADER *dos_head=(IMAGE_DOS_HEADER *)GetModuleHandle(NULL);
    PIMAGE_NT_HEADERS32 nt_head=(PIMAGE_NT_HEADERS32)((DWORD)dos_head+(DWORD)dos_head-
>e_lfanew);
    DWORD EP=(nt_head->OptionalHeader.AddressOfEntryPoint);
    CString str;
    str.Format("%x",EP);
    AfxMessageBox(str);
```

```
925
926     BYTE*OEP=(BYTE*)(nt_head->OptionalHeader.AddressOfEntryPoint+(DWORD)dos_head);
927     for(unsigned long index=0;index<20;index++){
928         if(OEP[index]==0xcc){
929             ExitProcess(0);
930         }
931     }
932
933 }
934 /************************************************************/
935 void terminateFunc()
936 {
937     AfxMessageBox("set_terminate指定的函数\n");
938     exit(0);
939 }
940 void CDetectODDlg::OnButton1()
941 {
942     // TODO: Add your control notification handler code here
943
944     set_terminate(terminateFunc);
945     try{
946         div(10,0);
947     }catch(int){
948         AfxMessageBox("仅捕获整型异常");
949     }catch(...){
950         terminate(); //所有其它异常
951     }
952     AfxMessageBox("啊哈");
953 }
954 //******************************************************
955
956 void CDetectODDlg::OnTrapFlag()
957 {
958     try{
959         _asm{
960             pushfd                  //触发单步异常
961             or      dword ptr [esp],100h   ;TF=1
962             popfd
963         }
964         AfxMessageBox("检测到OD");
965     }catch(...){
966         AfxMessageBox("没有OD");
967     }
968 }
969 //******************************************************
970 static bool isDebugged=1;
971 LONG WINAPI TopUnhandledExceptionFilter2(
972     struct _EXCEPTION_POINTERS *ExceptionInfo
973 )
974 {
975     _asm pushad
976     AfxMessageBox("回调函数");
977     lpSetUnhandledExceptionFilter((LPTOP_LEVEL_EXCEPTION_FILTER )lpOldHandler);
978     ExceptionInfo->ContextRecord->Eip=NewEip;
979     isDebugged=0;
980     _asm popad
981     return EXCEPTION_CONTINUE_EXECUTION;
982 }
983
984 void CDetectODDlg::OnGuardPages()
985 {
986     // TODO: Add your control notification handler code here
987
988     ULONG dwOldType;
989     DWORD dwPageSize;
990     LPVOID lpvBase;               // 获取内存的基地址
991     SYSTEM_INFO sSysInfo;         // 系统信息
992     GetSystemInfo(&sSysInfo);     // 获取系统信息
993     dwPageSize=sSysInfo.dwPageSize;     //系统内存页大小
994
995     lpSetUnhandledExceptionFilter =
(pSetUnhandledExceptionFilter)GetProcAddress(LoadLibrary(("kernel32.dll")),
996   "SetUnhandledExceptionFilter");
997     lpOldHandler=(DWORD)lpSetUnhandledExceptionFilter(TopUnhandledExceptionFilter2);
```

```
998
999   // 分配内存
1000    lpvBase = VirtualAlloc(NULL,dwPageSize,MEM_COMMIT,PAGE_READWRITE);
1001    if (lpvBase==NULL)  AfxMessageBox("内存分配失败");
1002    _asm{
1003        mov   NewEip,offset safe //方式二，更简单
1004        mov   eax,lpvBase
1005        push  eax
1006        mov   byte ptr [eax],0C3H //写一个 RETN 到保留内存，以便下面的调用
1007    }
1008    if(0==::VirtualProtect(lpvBase,dwPageSize,PAGE_EXECUTE_READ |
PAGE_GUARD,&dwOldType)){
1009        AfxMessageBox("执行失败");
1010    }
1011    _asm{
1012        pop   ecx
1013        call  ecx   //调用时压栈
1014safe:
1015        pop   ecx    //堆栈平衡，弹出调用时的压栈
1016    }
1017    if(1==isDebugged){
1018        AfxMessageBox("发现OD");
1019    }else{
1020        AfxMessageBox("没有OD");
1021    }
1022    VirtualFree(lpvBase,dwPageSize,MEM_DECOMMIT);
1023}
1024//******************************************************
1025static bool isDebuggedHBP=0;
1026LONG WINAPI TopUnhandledExceptionFilterHBP(
1027    struct _EXCEPTION_POINTERS *ExceptionInfo
1028)
1029{
1030    _asm pushad
1031    AfxMessageBox("回调函数被调用");
1032    ExceptionInfo->ContextRecord->Eip=NewEip;
1033    if(0!=ExceptionInfo->ContextRecord->Dr0||0!=ExceptionInfo->ContextRecord->Dr1||
1034        0!=ExceptionInfo->ContextRecord->Dr2||0!=ExceptionInfo->ContextRecord->Dr3)
1035        isDebuggedHBP=1;  //检测有无硬件断点
1036    ExceptionInfo->ContextRecord->Dr0=0; //禁用硬件断点，置0
1037    ExceptionInfo->ContextRecord->Dr1=0;
1038    ExceptionInfo->ContextRecord->Dr2=0;
1039    ExceptionInfo->ContextRecord->Dr3=0;
1040    ExceptionInfo->ContextRecord->Dr6=0;
1041    ExceptionInfo->ContextRecord->Dr7=0;
1042    ExceptionInfo->ContextRecord->Eip=NewEip; //转移到安全位置
1043    _asm popad
1044    return EXCEPTION_CONTINUE_EXECUTION;
1045}
1046
1047void CDetectODDlg::OnHardwarebreakpoint()
1048{
1049    // TODO: Add your control notification handler code here
1050
1051    lpSetUnhandledExceptionFilter =
(pSetUnhandledExceptionFilter)GetProcAddress(LoadLibrary(("kernel32.dll")),
1052  "SetUnhandledExceptionFilter");
1053    lpOldHandler=(DWORD)lpSetUnhandledExceptionFilter(TopUnhandledExceptionFilterHBP);
1054
1055    _asm{
1056        mov   NewEip,offset safe //方式二，更简单
1057        int   3
1058        mov   isDebuggedHBP,1 //调试时可能也不会触发异常去检测硬件断点
1059safe:
1060    }
1061    if(1==isDebuggedHBP){
1062        AfxMessageBox("发现OD");
1063    }else{
1064        AfxMessageBox("没有OD");
1065    }
1066}
1067//******************************************************
1068
1069void CDetectODDlg::OnCancel()
```

```
{
    // TODO: Add extra cleanup here
    CDialog::OnCancel();
}

void CAboutDlg::OnMypage()
{
    // TODO: Add your control notification handler code here
    ::ShellExecute(NULL,"open","http://ucooper.com",NULL,NULL,SW_SHOWNORMAL);
}

void CDetectODDlg::OnAbout()
{
    // TODO: Add your control notification handler code here
    CAboutDlg dlg;
    dlg.DoModal();
}

void CAboutDlg::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    CRect rect(60,20,100,100);
    if(rect.PtInRect(point)){
        SetClassLong(m_hWnd,GCL_HCURSOR,(LONG)(LoadCursor(NULL,IDC_HELP)));
    }else{
        SetClassLong(m_hWnd,GCL_HCURSOR,(LONG)(LoadCursor(AfxGetApp()->m_hInstance,IDC_ARROW)));
    }
    CDialog::OnMouseMove(nFlags, point);
}

BOOL CAboutDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // TODO: Add extra initialization here
    SetClassLong(m_hWnd,GCL_HICON,(LONG)(LoadIcon(AfxGetApp()->m_hInstance,MAKEINTRESOURCE(IDI_DOG))));
    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}

void CDetectODDlg::OnOK()
{
    // TODO: Add extra validation here

    CDialog::OnOK();
}

void CAboutDlg::OnComeon()
{
    // TODO: Add your control notification handler code here
    ::ShellExecute(NULL,"open","http://ucooper.com",NULL,NULL,SW_SHOWNORMAL);
}

void CAboutDlg::OnMyicon()
{
    // TODO: Add your control notification handler code here
    ::ShellExecute(NULL,"open","http://ucooper.com",NULL,NULL,SW_SHOWNORMAL);
}

void CDetectODDlg::OnMypage2()
{
    // TODO: Add your control notification handler code here
    ::ShellExecute(NULL,"open","http://ucooper.com",NULL,NULL,SW_SHOWNORMAL);
}

```