相信大家在 WINDOWS 系统上都遇到过想删除一个文件时却被提示“无法删除”的情况。我在初学电脑时遇到这种情况只能自认倒霉，重启之后再删除文件。学习了 WINDOWS 只是之后知道了在正常情况下（即不算文件被文件过滤驱动或者各种 API HOOK 保护的情况）遇到这个提示只有两种可能性：1.你没有删除这个文件的权限；2.有句柄在某个进程里被打开了。

解决第一种情况不需要编程，只要把以下文字保存成*.reg 文件并添加到注册表即可：

```
Windows Registry Editor Version 5.00

[HKEY_CLASSES_ROOT\*\shell\takeownership]
@="Take ownership"
"HasLUAShield"=""
"NoWorkingDirectory"=""

[HKEY_CLASSES_ROOT\*\shell\takeownership\command]
@="cmd.exe /c takeown /f \"%1\" && icacls \"%1\" /grant administrators:F"
"IsolatedCommand"="cmd.exe /c takeown /f \"%1\" && icacls \"%1\" /grant administrators:F"

[HKEY_CLASSES_ROOT\exefile\shell\takeownership]
@="Take ownership"
"HasLUAShield"=""
"NoWorkingDirectory"=""

[HKEY_CLASSES_ROOT\exefile\shell\takeownership\command]
@="cmd.exe /c takeown /f \"%1\" && icacls \"%1\" /grant administrators:F"
"IsolatedCommand"="cmd.exe /c takeown /f \"%1\" && icacls \"%1\" /grant administrators:F"

[HKEY_CLASSES_ROOT\dllfile\shell\takeownership]
@="Take ownership"
"HasLUAShield"=""
"NoWorkingDirectory"=""

[HKEY_CLASSES_ROOT\dllfile\shell\takeownership\command]
@="cmd.exe /c takeown /f \"%1\" && icacls \"%1\" /grant administrators:F"
"IsolatedCommand"="cmd.exe /c takeown /f \"%1\" && icacls \"%1\" /grant administrators:F"

[HKEY_CLASSES_ROOT\Directory\shell\takeownership]
@="Take ownership"
"HasLUAShield"=""
"NoWorkingDirectory"=""

[HKEY_CLASSES_ROOT\Directory\shell\takeownership\command]
@="cmd.exe /c takeown /f \"%1\" /r /d y && icacls \"%1\" /grant administrators:F /t"
```

```
"IsolatedCommand"="cmd.exe /c takeown /f \"%1\" /r /d y && icacls \"%1\" /grant administrators:F /t"
```

当你要删除一个文件而遇到“无法删除需要权限”的提示时，只要对着文件按下右键，选择『Take ownership』再删除文件即可。但是第二种情况就要通过编程解决了，这也就是本文的核心内容。要删除被打开的文件，比较好的方法是关闭此文件在其它进程里的句柄（直接解析文件系统修改标志位也可以，不过这个难度太大，而且不通用）。总体来说，步骤分为以下两步：1.枚举系统句柄表；2.获得所有和此文件有关的句柄并关闭。具体到代码级的思想，又可以分为以下几步：

1.调用 ZwQuerySystemInformation 的 16 功能号来枚举系统里的句柄
2.打开拥有此句柄的进程并把此句柄复制到自己的进程
3.用 ZwQueryObject 查询句柄的类型和名称
4.如果**发现此句柄的类型是文件句柄，名称和被锁定的文件一致**，就关闭此句柄
5.重复 2、3、4 步，直到遍历完系统里所有的句柄

代码如下：

```
VOID CloseFileHandle(char *szFileName)
{
    PVOID Buffer;
    ULONG BufferSize = 0x20000, rtl=0;
    NTSTATUS Status, qost=0;
    NTSTATUS ns = STATUS_SUCCESS;
    ULONG64 i=0;
    ULONG64 qwHandleCount;
    SYSTEM_HANDLE_TABLE_ENTRY_INFO *p;
    OBJECT_BASIC_INFORMATION BasicInfo;
    POBJECT_NAME_INFORMATION pNameInfo;
    ULONG ulProcessID;
    HANDLE hProcess;
    HANDLE hHandle;
    HANDLE hDupObj;
    CLIENT_ID cid;
    OBJECT_ATTRIBUTES oa;
    CHAR szFile[260]={0};
    Buffer=kmalloc(BufferSize);
    memset(Buffer,0,BufferSize);
    Status = ZwQuerySystemInformation(16, Buffer, BufferSize, 0);    //SystemHandleInformation
    while(Status == 0xC0000004) //STATUS_INFO_LENGTH_MISMATCH
    {
        kfree(Buffer);
        BufferSize = BufferSize * 2;
        Buffer=kmalloc(BufferSize);
        memset(Buffer,0,BufferSize);
```

```
        Status = ZwQuerySystemInformation(16, Buffer, BufferSize, 0);
    }
    if (!NT_SUCCESS(Status)) return;
    qwHandleCount=((SYSTEM_HANDLE_INFORMATION *)Buffer)->NumberOfHandles;
    p=(SYSTEM_HANDLE_TABLE_ENTRY_INFO *)((SYSTEM_HANDLE_INFORMATION *)Buffer)->Handles;
    //clear array
    memset(HandleInfo,0,1024*sizeof(HANDLE_INFO));
    //ENUM HANDLE PROC
    for(i=0;i<qwHandleCount;i++)
    {
        ulProcessID = (ULONG)p[i].UniqueProcessId;
        cid.UniqueProcess = (HANDLE)ulProcessID;
        cid.UniqueThread = (HANDLE)0;
        hHandle = (HANDLE)p[i].HandleValue;
        InitializeObjectAttributes( &oa ,NULL ,0 ,NULL ,NULL );
        ns = ZwOpenProcess( &hProcess ,PROCESS_DUP_HANDLE ,&oa ,&cid );
        if ( !NT_SUCCESS( ns ) )
        {
            KdPrint(( "ZwOpenProcess : Fail " ));
            continue;
        }
        ns = ZwDuplicateObject( hProcess ,hHandle ,NtCurrentProcess() ,&hDupObj ,
PROCESS_ALL_ACCESS ,0 ,DUPLICATE_SAME_ACCESS );
        if ( !NT_SUCCESS( ns ) )
        {
            KdPrint(( "ZwDuplicateObject : Fail " ));
            continue;
        }
        //get basic information
        ZwQueryObject( hDupObj ,ObjectBasicInformation ,&BasicInfo ,
sizeof( OBJECT_BASIC_INFORMATION ) ,NULL );
        //get name information
        pNameInfo = ExAllocatePoolWithTag( PagedPool ,1024 ,'ONON');
        RtlZeroMemory( pNameInfo ,1024 );
        qost=ZwQueryObject( hDupObj, ObjectNameInformation, pNameInfo, 1024, &rtl );
        //get information and close handle
        UnicodeStringToCharArray(&(pNameInfo->Name),szFile);
        ExFreePool( pNameInfo );
        ZwClose(hDupObj);
        ZwClose(hProcess);
        if(!strstr(_strlwr(szFile),szFileName))//这里只判断了文件名
        {
            PEPROCESS ep=LookupProcess((HANDLE)(p[i].UniqueProcessId));
            ForceCloseHandle(ep,p[i].HandleValue);
```

```
                ObDereferenceObject(ep);
            }
        }
}
```

接下来说说如何关闭其它进程里的句柄：

1.用 KeStackAttachProcess“依附”到目标进程
2.用 ObSetHandleAttributes 设置句柄为“可以关闭”
3.用 ZwClose 关闭句柄
4.用 KeUnstackDetachProcess 脱离“依附”目标进程

代码如下：

```
VOID ForceCloseHandle(PEPROCESS Process, ULONG64 HandleValue)
{
    HANDLE h;
    KAPC_STATE ks;
    OBJECT_HANDLE_FLAG_INFORMATION ohfi;
    if( Process==NULL )
        return;
    if( !MmIsAddressValid(Process) )
        return;
    KeStackAttachProcess(Process, &ks);
    h=(HANDLE)HandleValue;
    ohfi.Inherit=0;
    ohfi.ProtectFromClose=0;
    ObSetHandleAttributes(h, &ohfi, KernelMode);
    ZwClose(h);
    KeUnstackDetachProcess(&ks);
}
```

接下来说说测试步骤：

1.用『lockfile.exe』锁定文件『lockfile.xxx』
2.删除『lockfile.xxx』，会提示无法删除
3.加载『UnlockFile.sys』
4.再次删除『lockfile.xxx』则会成功

相关程序在 **WIN7 X64** 和 **WIN8 X64** 上测试通过（运行任何程序时都要以管理员权限运行）。在 WIN32 上可以用相同的方法，但是结构体的定义并不相同，对应的结构体需要你自己去寻找。

本文到此结束。示例代码在附件里。