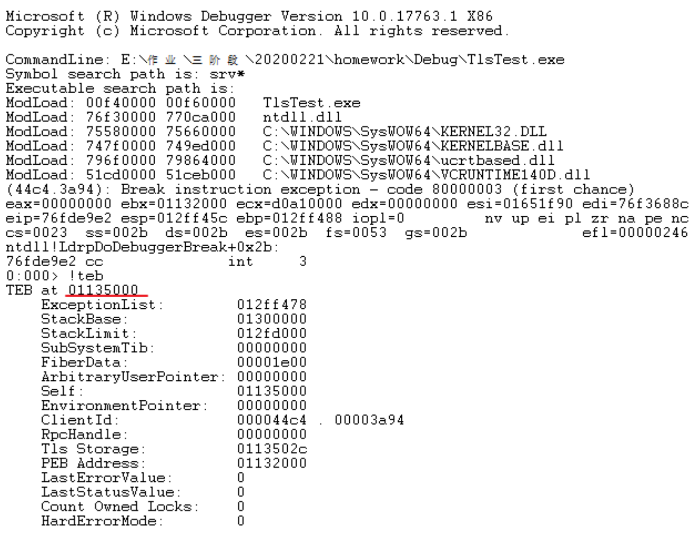
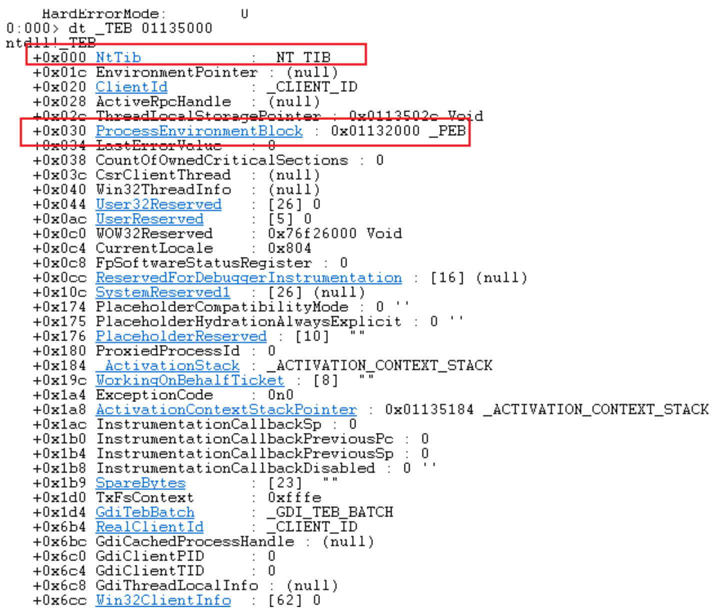
手工获取模块相关信息

- 获取TEB的位置并获得TEB



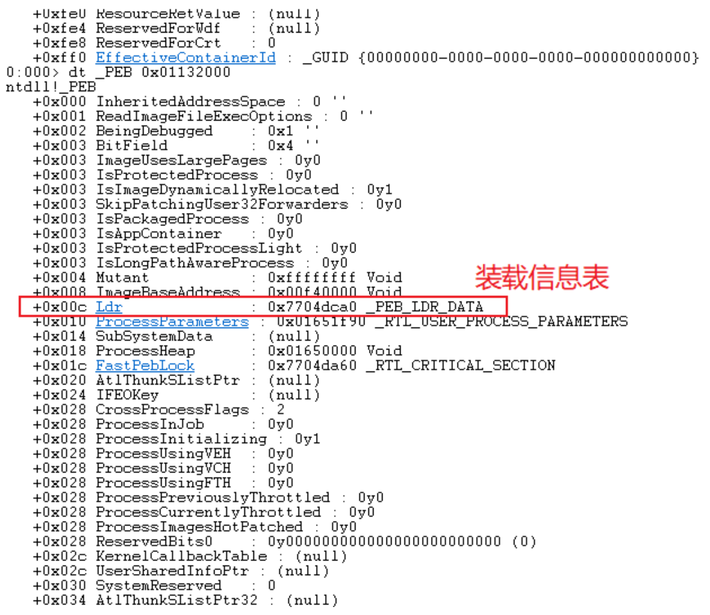
```
Microsoft (R) Windows Debugger Version 10.0.17763.1 X86
Copyright (c) Microsoft Corporation. All rights reserved.

CommandLine: E:\作 业 \三 阶 段 \20200221\homework\Debug\TlsTest.exe
Symbol search path is: srv*
Executable search path is:
ModLoad: 00f40000 00f60000   TlsTest.exe
ModLoad: 76f30000 770ca000   ntdll.dll
ModLoad: 75580000 75660000   C:\WINDOWS\SysWOW64\KERNEL32.DLL
ModLoad: 747f0000 749ed000   C:\WINDOWS\SysWOW64\KERNELBASE.dll
ModLoad: 796f0000 79864000   C:\WINDOWS\SysWOW64\ucrtbased.dll
ModLoad: 51cd0000 51ceb000   C:\WINDOWS\SysWOW64\VCRUNTIME140D.dll
(44c4.3a94): Break instruction exception - code 80000003 (first chance)
eax=00000000 ebx=01132000 ecx=d0a10000 edx=00000000 esi=01651f90 edi=76f3688c
eip=76fde9e2 esp=012ff45c ebp=012ff488 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
ntdll!LdrpDoDebuggerBreak+0x2b:
76fde9e2 cc              int     3
0:000> !teb
TEB at 01135000
    ExceptionList:        012ff478
    StackBase:            01300000
    StackLimit:           012fd000
    SubSystemTib:         00000000
    FiberData:            00001e00
    ArbitraryUserPointer: 00000000
    Self:                 01135000
    EnvironmentPointer:   00000000
    ClientId:             000044c4 . 00003a94
    RpcHandle:            00000000
    Tls Storage:          0113502c
    PEB Address:          01132000
    LastErrorValue:       0
    LastStatusValue:      0
    Count Owned Locks:    0
    HardErrorMode:        0
```

- 拿到TIB和PEB的位置



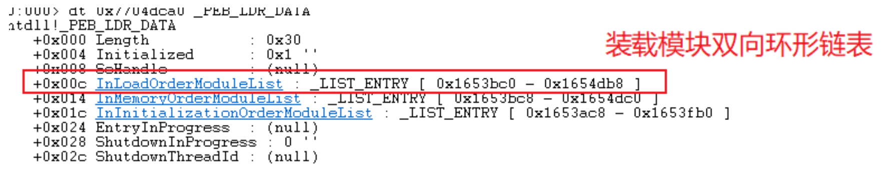
```
    HardErrorMode:        0
0:000> dt _TEB 01135000
ntdll!_TEB
   +0x000 NtTib            : _NT_TIB
   +0x01c EnvironmentPointer : (null)
   +0x020 ClientId         : _CLIENT_ID
   +0x028 ActiveRpcHandle  : (null)
   +0x02c ThreadLocalStoragePointer : 0x0113502c Void
   +0x030 ProcessEnvironmentBlock : 0x01132000 _PEB
   +0x034 LastErrorValue   : 0
   +0x038 CountOfOwnedCriticalSections : 0
   +0x03c CsrClientThread  : (null)
   +0x040 Win32ThreadInfo  : (null)
   +0x044 User32Reserved   : [26] 0
   +0x0ac UserReserved     : [5] 0
   +0x0c0 WOW32Reserved    : 0x76f26000 Void
   +0x0c4 CurrentLocale    : 0x804
   +0x0c8 FpSoftwareStatusRegister : 0
   +0x0cc ReservedForDebuggerInstrumentation : [16] (null)
   +0x10c SystemReserved1  : [26] (null)
   +0x174 PlaceholderCompatibilityMode : 0 ''
   +0x175 PlaceholderHydrationAlwaysExplicit : 0 ''
   +0x176 PlaceholderReserved : [10]  ""
   +0x180 ProxiedProcessId : 0
   +0x184 _ActivationStack : _ACTIVATION_CONTEXT_STACK
   +0x19c WorkingOnBehalfTicket : [8]  ""
   +0x1a4 ExceptionCode    : 0n0
   +0x1a8 ActivationContextStackPointer : 0x01135184 _ACTIVATION_CONTEXT_STACK
   +0x1ac InstrumentationCallbackSp : 0
   +0x1b0 InstrumentationCallbackPreviousPc : 0
   +0x1b4 InstrumentationCallbackPreviousSp : 0
   +0x1b8 InstrumentationCallbackDisabled : 0 ''
   +0x1b9 SpareBytes       : [23]  ""
   +0x1d0 TxFsContext      : 0xfffe
   +0x1d4 GdiTebBatch      : _GDI_TEB_BATCH
   +0x6b4 RealClientId     : _CLIENT_ID
   +0x6bc GdiCachedProcessHandle : (null)
   +0x6c0 GdiClientPID     : 0
   +0x6c4 GdiClientTID     : 0
   +0x6c8 GdiThreadLocalInfo : (null)
   +0x6cc Win32ClientInfo  : [62] 0
```

- 查询PEB，获得装载信息表`_PEB_LDR_DATA`



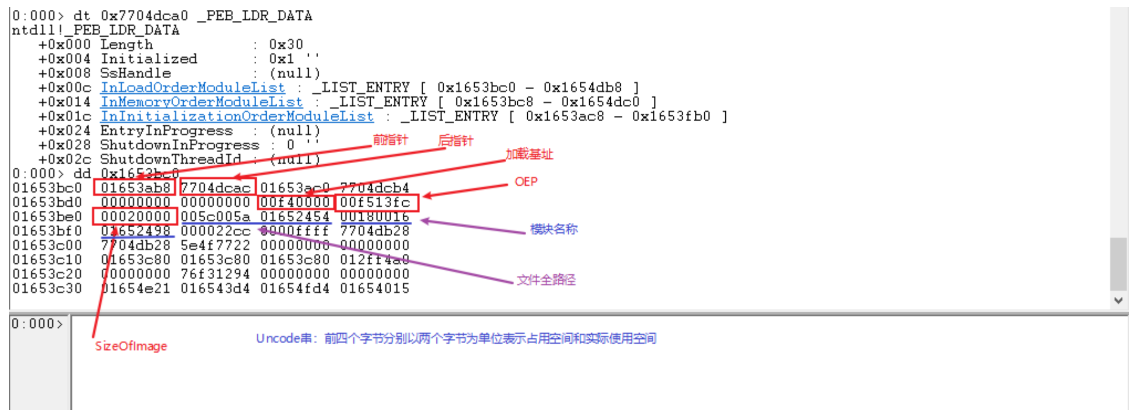
```
   +0xfe0 ResourceRetValue : (null)
   +0xfe4 ReservedForWdf   : (null)
   +0xfe8 ReservedForCrt   : 0
   +0xff0 EffectiveContainerId : _GUID {00000000-0000-0000-0000-000000000000}
0:000> dt _PEB 0x01132000
ntdll!_PEB
   +0x000 InheritedAddressSpace : 0 ''
   +0x001 ReadImageFileExecOptions : 0 ''
   +0x002 BeingDebugged    : 0x1 ''
   +0x003 BitField         : 0x4 ''
   +0x003 ImageUsesLargePages : 0y0
   +0x003 IsProtectedProcess : 0y0
   +0x003 IsImageDynamicallyRelocated : 0y1
   +0x003 SkipPatchingUser32Forwarders : 0y0
   +0x003 IsPackagedProcess : 0y0
   +0x003 IsAppContainer   : 0y0
   +0x003 IsProtectedProcessLight : 0y0
   +0x003 IsLongPathAwareProcess : 0y0
   +0x004 Mutant           : 0xffffffff Void
   +0x008 ImageBaseAddress : 0x00f40000 Void
   +0x00c Ldr              : 0x7704dca0 _PEB_LDR_DATA
   +0x010 ProcessParameters : 0x01651f90 _RTL_USER_PROCESS_PARAMETERS
   +0x014 SubSystemData    : (null)
   +0x018 ProcessHeap      : 0x01650000 Void
   +0x01c FastPebLock      : 0x7704da60 _RTL_CRITICAL_SECTION
   +0x020 AtlThunkSListPtr : (null)
   +0x024 IFEOKey          : (null)
   +0x028 CrossProcessFlags : 2
   +0x028 ProcessInJob     : 0y0
   +0x028 ProcessInitializing : 0y1
   +0x028 ProcessUsingVEH  : 0y0
   +0x028 ProcessUsingVCH  : 0y0
   +0x028 ProcessUsingFTH  : 0y0
   +0x028 ProcessPreviouslyThrottled : 0y0
   +0x028 ProcessCurrentlyThrottled : 0y0
   +0x028 ProcessImagesHotPatched : 0y0
   +0x028 ReservedBits0    : 0y000000000000000000000000 (0)
   +0x02c KernelCallbackTable : (null)
   +0x02c UserSharedInfoPtr : (null)
   +0x030 SystemReserved   : 0
   +0x034 AtlThunkSListPtr32 : (null)
```

- 装载信息表



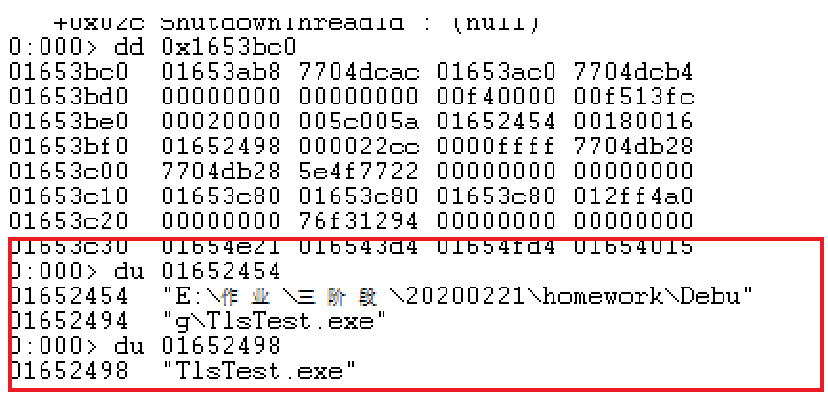
```
0:000> dt 0x7704dca0 _PEB_LDR_DATA
ntdll!_PEB_LDR_DATA
   +0x000 Length           : 0x30
   +0x004 Initialized      : 0x1 ''
   +0x008 SsHandle         : (null)
   +0x00c InLoadOrderModuleList : _LIST_ENTRY [ 0x1653bc0 - 0x1654db8 ]
   +0x014 InMemoryOrderModuleList : _LIST_ENTRY [ 0x1653bc8 - 0x1654dc0 ]
   +0x01c InInitializationOrderModuleList : _LIST_ENTRY [ 0x1653ac8 - 0x1653fb0 ]
   +0x024 EntryInProgress  : (null)
   +0x028 ShutdownInProgress : 0 ''
   +0x02c ShutdownThreadId : (null)
```

- 模块链表


前指针
后指针
加载基址
OEP
模块名称
文件全路径
SizeOfImage
Uncode串：前四个字节分别以两个字节为单位表示占用空间和实际使用空间

```
0:000> dt 0x7704dca0 _PEB_LDR_DATA
ntdll!_PEB_LDR_DATA
   +0x000 Length           : 0x30
   +0x004 Initialized      : 0x1 ''
   +0x008 SsHandle         : (null)
   +0x00c InLoadOrderModuleList : _LIST_ENTRY [ 0x1653bc0 - 0x1654db8 ]
   +0x014 InMemoryOrderModuleList : _LIST_ENTRY [ 0x1653bc8 - 0x1654dc0 ]
   +0x01c InInitializationOrderModuleList : _LIST_ENTRY [ 0x1653ac8 - 0x1653fb0 ]
   +0x024 EntryInProgress  : (null)
   +0x028 ShutdownInProgress : 0 ''
   +0x02c ShutdownThreadId : (null)
0:000> dd 0x1653bc0
01653bc0  01653ab8 7704dcac 01653ac0 7704dcb4
01653bd0  00000000 00000000 00f40000 00f513fc
01653be0  00020000 005c005a 01652454 00180016
01653bf0  01652498 000022cc 0000ffff 7704db28
01653c00  7704db28 5e4f7722 00000000 00000000
01653c10  01653c80 01653c80 01653c80 012ff4a0
01653c20  00000000 76f31294 00000000 00000000
01653c30  01654e21 016543d4 01654fd4 01654015

0:000>
```

```
// 所使用的Unicode串
typedef struct _UNICODE_STRING {
  USHORT Length;
  USHORT MaximumLength;
  PWSTR  Buffer;
} UNICODE_STRING, *PUNICODE_STRING;
```



```
   +0x02c ShutdownThreadId : (null)
0:000> dd 0x1653bc0
01653bc0  01653ab8 7704dcac 01653ac0 7704dcb4
01653bd0  00000000 00000000 00f40000 00f513fc
01653be0  00020000 005c005a 01652454 00180016
01653bf0  01652498 000022cc 0000ffff 7704db28
01653c00  7704db28 5e4f7722 00000000 00000000
01653c10  01653c80 01653c80 01653c80 012ff4a0
01653c20  00000000 76f31294 00000000 00000000
01653c30  01654e21 016543d4 01654fd4 01654015
0:000> du 01652454
01652454  "E:\作 业 \三 阶 段 \20200221\homework\Debu"
01652494  "g\TlsTest.exe"
0:000> du 01652498
01652498  "TlsTest.exe"
```

代码示例

```
DWORD GetLoadOrderModuleList()
{
    DWORD pointer = 0;
    __asm {
        push eax
        xor eax, eax

        mov eax, fs:[0x30] // 获取PEB

        // 获取装载信息表
        lea eax, [eax + 0x0c]
        mov eax, dword ptr [eax]

        // 获取模块链表
        lea eax, [eax + 0x0c]
        mov eax, dword ptr[eax]

        mov pointer, eax

        pop eax
    }

    return pointer;
}


HMODULE WINAPI MyGetModuleHandle(LPCTSTR lpModuleName)
{
    DWORD module_list = GetLoadOrderModuleList();
    ModuleItem *item = (ModuleItem *)module_list;
    while(item->pointer.Flink != (_LIST_ENTRY *)module_list) {
        // 遍历
        //printf("%ls\n", item->name.Buffer);
        DWORD length = wcslen(lpModuleName);
        if(length == wcslen(item->name.Buffer)) {
            bool found = true;
            for(DWORD i = 0; i < length; i++) {
                if(towupper(lpModuleName[i]) != towupper(item->name.Buffer[i]))
{
                    found = false;
                    break;
                }
            }
            if(found) {
                return (HMODULE)item->base;
            }
        }

        item = (ModuleItem *)item->pointer.Flink;
    }
    return 0;
}
```

```
DWORD WINAPI MyGetModuleFileName(HMODULE hModule, LPTSTR lpFilename, DWORD
nSize)
{
    memset(lpFilename, 0, sizeof(TCHAR) * nSize);
    DWORD module_list = GetLoadOrderModuleList();
    ModuleItem *item = (ModuleItem *)module_list;
    while (item->pointer.Flink != (_LIST_ENTRY *)module_list) {
        // 遍历
        //printf("%ls\n", item->name.Buffer);
        if (hModule == (HMODULE)item->base) {
            int count = min(wcslen(item->path.Buffer), nSize);
            wcsncpy(lpFilename, item->path.Buffer, count);
            return count;
        }

        item = (ModuleItem *)item->pointer.Flink;
    }
    return 0;
}


DWORD WINAPI MyGetModuleBaseName(HMODULE hModule, LPTSTR lpFilename, DWORD
nSize)
{
    memset(lpFilename, 0, sizeof(TCHAR) * nSize);
    DWORD module_list = GetLoadOrderModuleList();
    ModuleItem *item = (ModuleItem *)module_list;
    while (item->pointer.Flink != (_LIST_ENTRY *)module_list) {
        // 遍历
        //printf("%ls\n", item->name.Buffer);
        if(hModule == (HMODULE)item->base) {
            int count = min(wcslen(item->name.Buffer), nSize);
            wcsncpy(lpFilename, item->name.Buffer, count);
            return count;
        }

        item = (ModuleItem *)item->pointer.Flink;
    }
    return 0;
}
```