# 汇编语言与逆向技术实验报告

## Lab4-peviewer

学号：2112060　　姓名：孙蕗　　专业：信息安全

## 一、 实验目的

1、　熟悉 PE 文件结构；

2、　使用 Windows API 函数读取文件内容

## 二、 实验原理

### （1）PE 文件结构

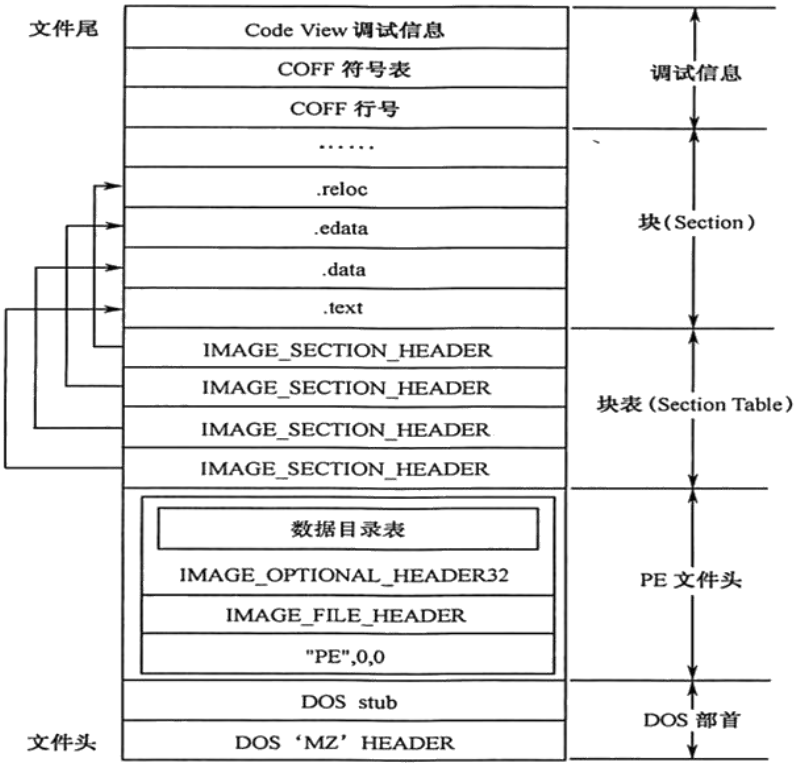
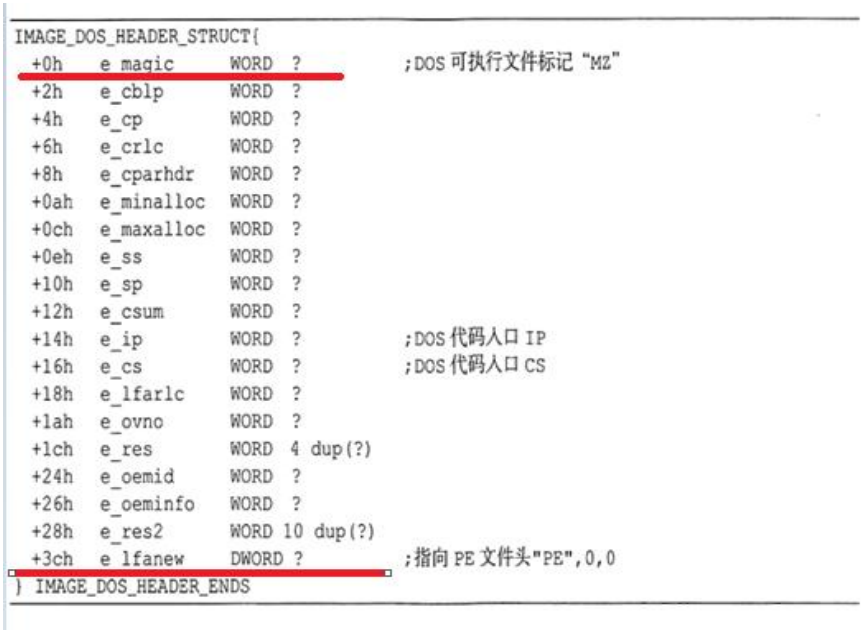



图 1 PE 文件结构

PE 文件结构如图 1 所示。二进制 PE 文件包括：DOS 部首、PE 文件头、块表（Section Table）、块（Section）、调试信息 5 个部分。

DOS 部首是 DOS 系统的残留内容，目的是防止 Windows 系统的可执行程序在 DOS 系统上执行时导致 DOS 系统崩溃。DOS 部首是

一段 DOS 程序，输出一段提示信息，说明程序只能运行在 Windows 系统上，不能运行在 DOS 系统上。



```
IMAGE_DOS_HEADER_STRUCT{
  +0h    e_magic     WORD  ?          ;DOS 可执行文件标记 "MZ"
  +2h    e_cblp      WORD  ?
  +4h    e_cp        WORD  ?
  +6h    e_crlc      WORD  ?
  +8h    e_cparhdr   WORD  ?
  +0ah   e_minalloc  WORD  ?
  +0ch   e_maxalloc  WORD  ?
  +0eh   e_ss        WORD  ?
  +10h   e_sp        WORD  ?
  +12h   e_csum      WORD  ?
  +14h   e_ip        WORD  ?          ;DOS 代码入口 IP
  +16h   e_cs        WORD  ?          ;DOS 代码入口 CS
  +18h   e_lfarlc    WORD  ?
  +1ah   e_ovno      WORD  ?
  +1ch   e_res       WORD  4 dup(?)
  +24h   e_oemid     WORD  ?
  +26h   e_oeminfo   WORD  ?
  +28h   e_res2      WORD 10 dup(?)
  +3ch   e_lfanew    DWORD ?          ;指向 PE 文件头"PE",0,0
} IMAGE_DOS_HEADER_ENDS
```

图 2 DOS 头的数据结构

PE 文件头记录了各种文件的装载信息，有映像的基地址（ImageBase）、程序的入口地址（EntryPoint）、数据块、编译时间、运行平台、数据目录表等信息。PE 文件头包括 Signature、FileHeader、OptionalHeader 三部分，数据结构如下所示：

IMAGE_NT_HEADERS STRUCT

+0h Signature DWORD ?

+4h FileHeader IMAGE_FILE_HEADER <>

+18h OptionalHeader IMAGE_OPTIONAL_HEADER32 <>

IMAGE_NT_HEADERS ENDS

**Signature** 的定义是 IMAGE_NT_HEADER，值为 00004550h

**FileHeader** 的数据结构如下所示：

IMAGE_FILE_HEADER STRUCT

| | | |
|---|---|---|
| +04h Machine | WORD | ? |
| +06h NumberOfSections | WORD | ? |
| +08h TimeDateStamp | DWORD | ? |
| +0Ch PointerToSymbolTable | DWORD | ? |
| +10h NumberOfSymbols | DWORD | ? |
| +14h SizeOfOptionalHeader | WORD | ? |
| +16h Characteristics | WORD | ? |

**OptionalHeader** 的数据结构如下所示：

IMAGE_OPTIONAL_HEADER STRUCT

| | | | |
|---|---|---|---|
| +18h | Magic | WORD | ? |
| +1Ah | MajorLinkerVersion | BYTE | ? |
| +1Bh | MinorLinkerVersion | BYTE | ? |
| +1Ch | SizeOfCode | DWORD | ? |
| +20h | SizeOfInitializedData | DWORD | ? |
| +24h | SizeOfUninitializedData | DWORD | ? |
| +28h | AddressOfEntryPoint | DWORD | ? |
| +2Ch | BaseOfCode | DWORD | ? |
| +30h | BaseOfData | DWORD | ? |
| +34h | ImageBase | DWORD | ? |
| +38h | SectionAlignment | DWORD | ? |

| | | | |
|---|---|---|---|
| +3Ch | FileAlignment | DWORD | ? |
| +40h | MajorOperatingSystemVersion | WORD | ? |
| +42h | MinorOperatingSystemVersion | WORD | ? |
| +44h | MajorImageVersion | WORD | ? |
| +46h | MinorImageVersion | WORD | ? |
| +48h | MajorSubsystemVersion | WORD | ? |
| +4Ah | MinorSubsystemVersion | WORD | ? |
| +4Ch | Win32VersionValue | DWORD | ? |
| +50h | SizeOfImage | DWORD | ? |
| +54h | SizeOfHeaders | DWORD | ? |
| +58h | CheckSum | DWORD | ? |
| +5Ch | Subsystem | WORD | ? |
| +5Eh | DllCharacteristics | WORD | ? |
| +60h | SizeOfStackReserve | DWORD | ? |
| +64h | SizeOfStackCommit | DWORD | ? |
| +68h | SizeOfHeapReserve | DWORD | ? |
| +6Ch | SizeOfHeapCommit | DWORD | ? |
| +70h | LoaderFlags | DWORD | ? |
| +74h | NumberOfRvaAndSizes | DWORD | ? |
| +78h | DataDirectory | | |

[IMAGE_NUMBEROF_DIRECTORY_ENTRIES]

IMAGE_DATA_DIRECTORY <>

IMAGE_OPTIONAL_HEADER ENDS

块表（Section Table）描述代码块、数据块、资源块等不同数据块的文件和内存的映射，数据块的各种属性。

块（Section）分别存储了程序的代码、数据、资源等信息。

**（2） Windows 文件读操作**

读一个文件用到的 Windows API 函数有 CreateFile、SetFilePointer、ReadFile、CloseHandle。

**CreateFile** 的 MSDN 文档地址 https://docs.microsoft.com/en-us/previous-versions/aa914735(v=msdn.10)，函数的原型如下：

```
HANDLE CreateFile(
  LPCTSTR lpFileName,
  DWORD dwDesiredAccess,
  DWORD dwShareMode,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  DWORD dwCreationDisposition,
  DWORD dwFlagsAndAttributes,
  HANDLE hTemplateFile
    );
```

CreateFile 在 MASM 汇编语言中的应用实例如图 3 所示。

```
invoke CreateFile,ADDR buf2,\
                  GENERIC_READ,\
                  FILE_SHARE_READ,\
                  0,\
                  OPEN_EXISTING,\
                  FILE_ATTRIBUTE_ARCHIVE,\
                  0
MOV hfile, EAX
invoke SetFilePointer, hfile, 0, 0, FILE_BEGIN
invoke ReadFile, hfile, ADDR buf3, 4000, 0, 0
MOV EAX, DWORD PTR buf3
invoke dw2hex, EAX, ADDR buf4
invoke StdOut, ADDR buf4
invoke CloseHandle, hfile
```

图 3 MASM 汇编中调用 CreateFile、SetFilePointer、ReadFile、CloseHandle 的示例

**SetFilePointer** 函数的 MSDN 文档地址

https://docs.microsoft.com/en-us/previous-versions/aa911934(v%3dmsdn.

10)，函数原型如下：

```
DWORD SetFilePointer(
  HANDLE hFile,
  LONG lDistanceToMove,
  PLONG lpDistanceToMoveHigh,
  DWORD dwMoveMethod
    );
```

SetFilePointer 函数的 MASM 调用示例如图 3 所示。

**ReadFile** 函数的 MSDN 文档地址

https://docs.microsoft.com/en-us/previous-versions/aa914377(v%3dmsdn.

10)，函数原型如下：

```
BOOL ReadFile(
  HANDLE hFile,
  LPVOID lpBuffer,
  DWORD nNumberOfBytesToRead,
  LPDWORD lpNumberOfBytesRead,
  LPOVERLAPPED lpOverlapped
);
```

ReadFile 函数的 MASM 调用示例如图 3 所示。

**CloseHandle** 函数的 MSDN 文档地址

https://docs.microsoft.com/en-us/previous-versions/aa914720(v%3dmsdn.

10)，函数原型如下：

```
BOOL CloseHandle(
  HANDLE hObject
);
```

CloseHandle 函数的 MASM 调用示例如图 3 所示。

## 三、 实验环境

Windows 操作系统，MASM32 编译环境。

## 四、 实验内容：

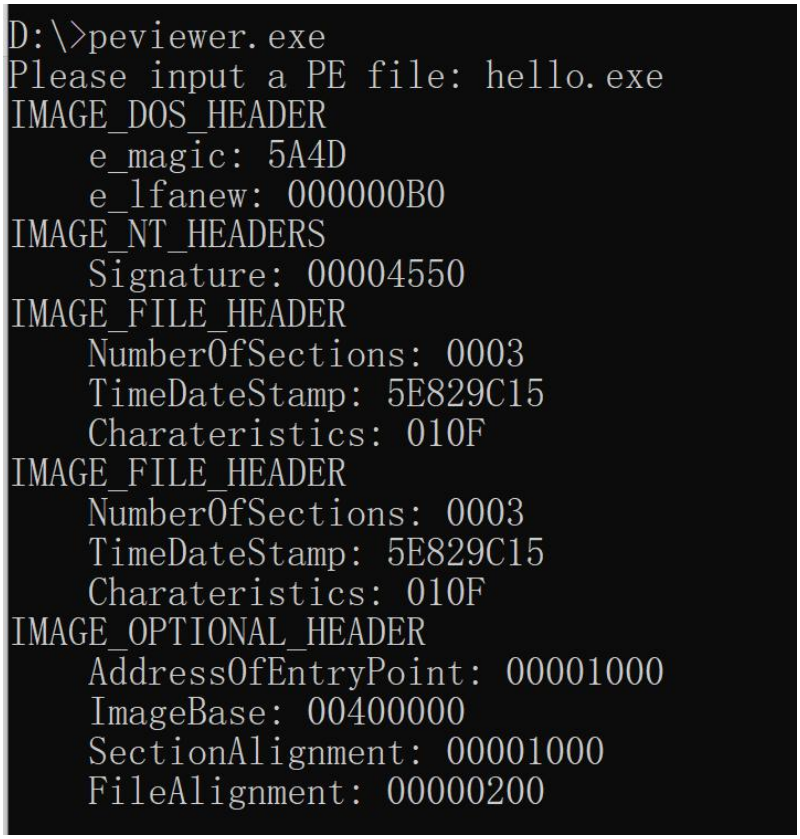



图 4 peviewer 实验演示

（1）输入 PE 文件的文件名，peviewer 程序调用 Windows API 函数，打开指定的 PE 文件；

（2）从文件的头部开始，读取 IMAGE_DOS_HEADER 结构中的 e_magic 和 e_lfanew 字段的值，按照实验演示的方式输出到命令行窗口；

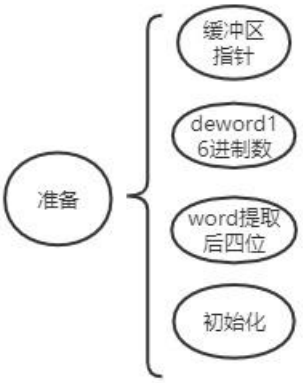
（3）继续读取 PE 文件的 IMAGE_NT_HEADER 结构中的 Signature 字段的值，按照实验演示的方式输出到命令行窗口；

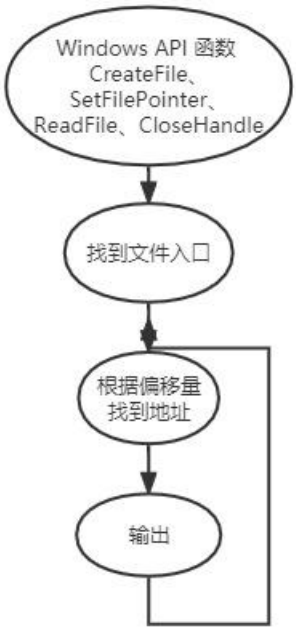
（4）继续读取 IMAGE_NT_HEADER 结构中的 IMAGE_FILE_HEADER 结构，从中读取出字段

NumberOfSections、TimeDateStamp、Characteristics 的值，按照实验演示的方式输出到命令行窗口；

（5）继续读取 IMAGE_NT_HEADER 结构中的 IMAGE_OPTIONAL_HEADER 结构，从中读取字段 AddressOfEntryPoint、ImageBase、SectionAlignment、FileAlignment 的值，按照实验演示的方式输出到命令行窗口；

五、 peviewer 程序的设计说明和控制流图




1）Dos 头读取

从文件头开始位置，读取固定 64 个字节为 dos 头。保存并且解析结构内容。

2）PE 头读取

根据 Dos 头的 e_lfanew 域的内容（新 exe 头在文件中的地址），定位文件指针，读取 24 字节的 PE 文件头。从 PE 文件头中解析出可选头的大小，按照该字节数在后序的代码中读取 PE 可选头，并且进行解析。

3）段表读取

文件可选域中获得段的的数量。按照段的数量读取 40 字节乘该数量的字节数，分别为各个段表赋值。

4）读取输入表

从文件的可选域目录表的数组中，找到 1 号元素，获得输入表的虚地址（内存偏移地址）属性。将该内存偏移地址转换为文件偏移地址，并将文件读写指针定位到此。从该起始地址开始，读取第一个输入表的结构体。完成读取后查看输入表的 FirstThunk 属性，如果为零则跳出循环，输入表查找完毕。

5）读取输出表

从文件可选域中的目录表数组中，找到 0 号元素，获得输出表的虚地址（内存偏移地址）属性。将内存偏移值转换为文件偏移值，从该文件起始地址，抓取输出表的结构体，完成赋值工作。输出表数量加一，输出表数量也只有 1，其他与输入表的读取过程完全相同。

6）读取资源表

根据 PE 文件资源表的布局情况，我们发现其整体结构是一个树形结构的，这样就给查找过程带来了麻烦。所以，对数据结点的查找应该是一个递归的过程，对目录项结点层层分析，直到找到标记值不为零的结点。

## 六、 peviewer.asm 的源代码和注释

```
.386
.model flat,stdcall
option casemap :none
include \masm32\include\windows.inc
include \masm32\include\masm32.inc
```

```
include \masm32\macros\macros.asm
include \masm32\include\kernel32.inc
includelib \masm32\lib\masm32.lib
includelib \masm32\lib\kernel32.lib

.data
;que 输出的提示字符串
que BYTE "Please input a PE file: ",0
que11 BYTE "IMAGE_DOS_HEADER",0Ah,0Dh,0
que12 BYTE "        e_magic:",0
que13 BYTE "        e_lfanew:",0
que21 BYTE "IMAGE_NT_HEADERS",0Ah,0Dh,0
que22 BYTE "        Signature:",0
que31 BYTE "IMAGE_FILE_HEADER",0Ah,0Dh,0
que32 BYTE "        NumberOfSections:",0
que33 BYTE "        TimeDateStamp:",0
que34 BYTE "        Characteristics:",0
que41 BYTE "IMAGE_OPTIONAL_HEADER",0Ah,0Dh,0
que42 BYTE "        AddressOfEntryPoint:",0
que43 BYTE "        ImageBase:",0
que44 BYTE "        SectionAlignment:",0
que45 BYTE "        FileAlignment:",0
```

```
buf3 DWORD 4000 DUP(0)      ;缓冲区指针
buf4 DWORD 4000 DUP(0)      ;转成十六进制
buf5 WORD 4000 DUP (0)      ;存储后四位
ans BYTE 400 DUP(0),0
hfile DWORD 0,0        ;文件句柄
endl BYTE 0Ah,0Dh,0              ;换行
temp DWORD 0,0              ;存 ecx 的值
temp1 DWORD 0,0              ;定位指针的初始位置，是 Dos 头还是 PE 头

.code
Output PROC              ;读取相应位置的内容并转化成二进制
;edx 为偏移量
    mov esi,OFFSET buf3
    add esi,edx          ;把偏移量加上
    add esi,temp1      ;DOS 头还是 NT 头
    mov eax,DWORD PTR[esi]
    mov ebx,eax
    invoke dw2hex_ex,eax,addr buf4              ;dw2hex_ex 用于查看寄存器 eax 的 16 进制信息
    mov ecx,temp
```

```
        .if ecx==8
        invoke StdOut,addr buf4      ;转成十六进制
        .else
        mov ax,WORD PTR [buf4+4]
        mov buf5,ax        ;存储后四位
        invoke StdOut,addr buf5

        mov ax,WORD PTR [buf4+6]
        mov buf5,ax
        invoke StdOut,addr buf5
        .endif
        invoke StdOut,addr endl
ret
Output ENDP

start:
        invoke StdOut,addr que    ;输出请求
        invoke StdIn,addr ans,400        ;输入 exe 名称(最好输入路径)
;打开文件
        invoke CreateFile,addr ans,\              ;打开文件，将句柄保存到 eax
                                GENERIC_READ,\
                                FILE_SHARE_READ,\
```

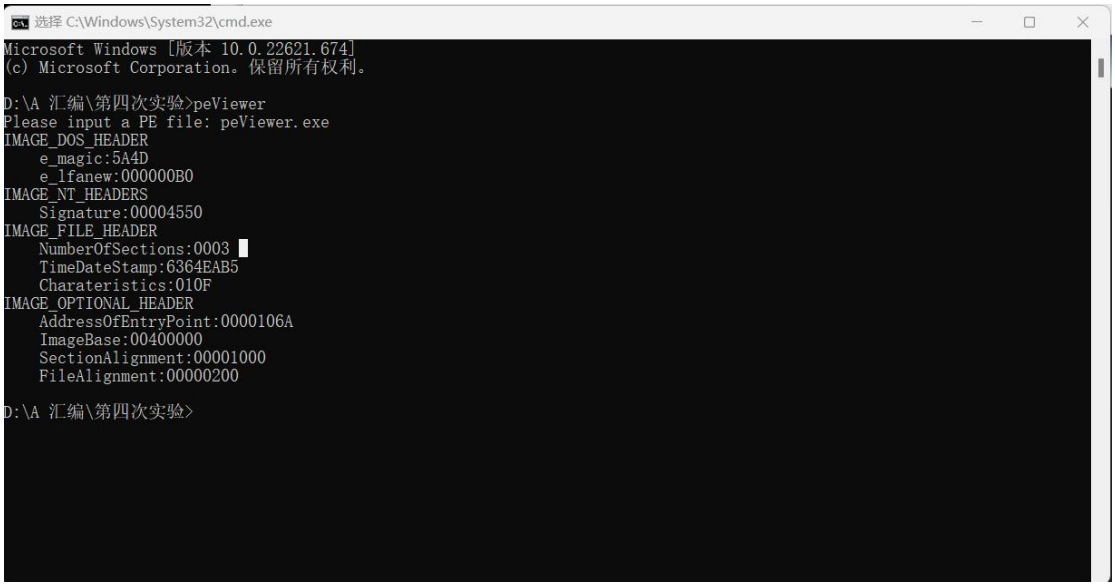
```
                                0,\
                                OPEN_EXISTING,\
                                FILE_ATTRIBUTE_ARCHIVE,\
                                0
        mov hfile,eax      ;保存文件句柄
        invoke SetFilePointer,hfile,0,0,FILE_BEGIN;调用 SetFilePointer
        invoke ReadFile,hfile,addr buf3,4000,0,0      ;调用 ReadFile,
将文件入口设置为 buf3
        mov esi,OFFSET buf3;文件入口
        ;buf3 的地址是 Dos 部首的头
        invoke StdOut,addr que11      ; "IMAGE_DOS_HEADER"
        invoke StdOut,addr que12       ; e_magic

;e_magic
        mov edx,0      ;edx 为偏移量
        mov temp1,edx
        mov ecx,4        ;word
        mov temp,ecx
        invoke Output

        invoke StdOut,addr que13    ; e_lfanew
;e_lfanew
```

```
        mov edx,3ch         ;edx 偏移量
        mov ecx,8      ;dword
        mov temp,ecx
        invoke Output

        invoke StdOut,addr que21    ; IMAGE_NT_HEADERS
        invoke StdOut,addr que22 ;Signature
        mov temp1,ebx
;Signature
        mov edx,0       ;edx 偏移量，dword
        invoke Output

        invoke StdOut,addr que31       ;IMAGE_FILE_HEADER
        invoke StdOut,addr que32      ;NumberOfSections
;NumberOfSections
        mov edx,6h         ;edx 偏移量
        mov ecx,4            ;word
        mov temp,ecx
        invoke Output

        invoke StdOut,addr que33       ;TimeDateStamp
;TimeDateStamp
```

```
        mov edx,8h        ;edx 偏移量
        mov ecx,8         ;dword
        mov temp,ecx
        invoke Output

        invoke StdOut,addr que34          ;Characteristics
;Characteristics
        mov edx,16h    ;edx 偏移量
        mov ecx,4         ;word
        mov temp,ecx
        invoke Output

        invoke StdOut,addr que41        ;IMAGE_OPTIONAL_HEADER
        invoke StdOut,addr que42        ;AddressOfEntryPoint
;AddressOfEntryPoint
        mov edx,28h      ;edx 偏移量
        mov ecx,8
        mov temp,ecx
        invoke Output

        invoke StdOut,addr que43        ;ImageBase
;ImageBase
```

```
        mov edx,34h      ;edx 偏移量，dword
        invoke Output

        invoke StdOut,addr que44    ;SectionAlignment
;SectionAlignment
        mov edx,38h       ;edx 偏移量，dword
        invoke Output

        invoke StdOut,addr que45      ;FileAlignment
;FileAlignment
        mov edx,3ch       ;edx 偏移量，dword
        invoke Output
invoke CloseHandle,hfile;调用 CloseHandle
invoke ExitProcess,0
end start
```

七、 peviewer.exe 运行截图


选择 C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.22621.674]
(c) Microsoft Corporation。保留所有权利。
D:\A 汇编\第四次实验>peViewer
Please input a PE file: peViewer.exe
IMAGE_DOS_HEADER
e_magic:5A4D
e_lfanew:000000B0
IMAGE_NT_HEADERS
Signature:00004550
IMAGE_FILE_HEADER
NumberOfSections:0003
TimeDateStamp:6364EAB5
Charateristics:010F
IMAGE_OPTIONAL_HEADER
AddressOfEntryPoint:0000106A
ImageBase:00400000
SectionAlignment:00001000
FileAlignment:00000200
D:\A 汇编\第四次实验>