**汇编语言与逆向技术实验报告**

**Lab4-peviewer**

**学号：2112492 姓名：刘修铭 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**

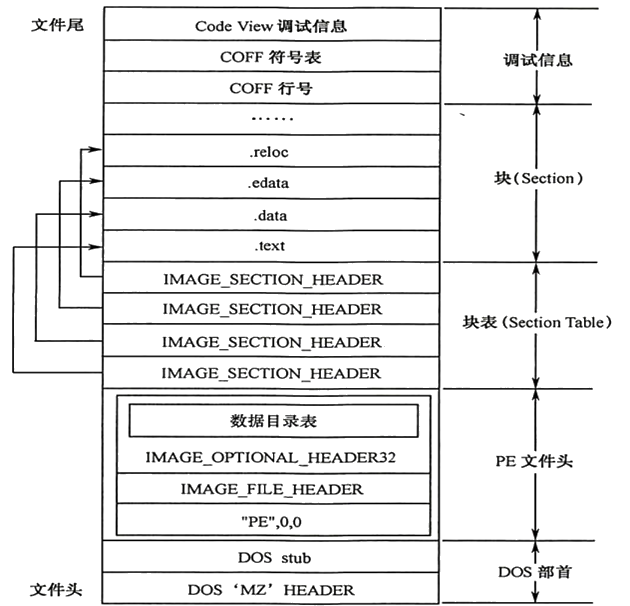


图 1 PE文件结构

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。

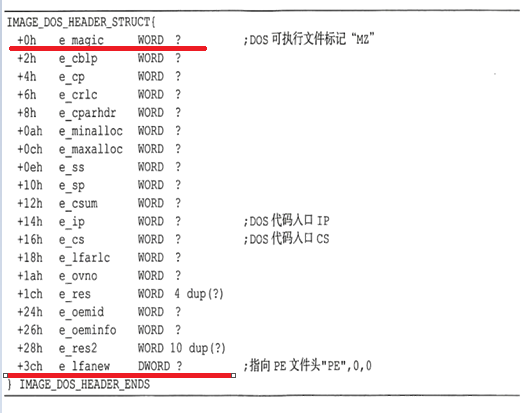


图 2 DOS头的数据结构

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。

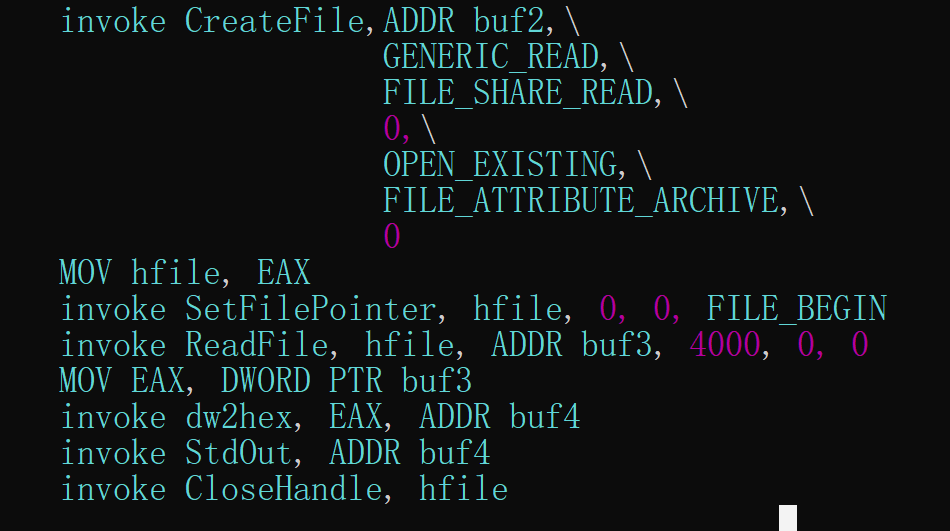


图 3 MASM汇编中调用CreateFile、SetFilePointer、ReadFile、CloseHandle的示例

**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**

MASM32编译环境

Windows操作系统

**四、实验内容**

（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、过程说明**

**1. 编辑：编写汇编程序peviewer.asm。**

* 流程：

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；

* 设计说明：

1.调用CreateFile函数，利用用户输入的文件名打开相应文件；

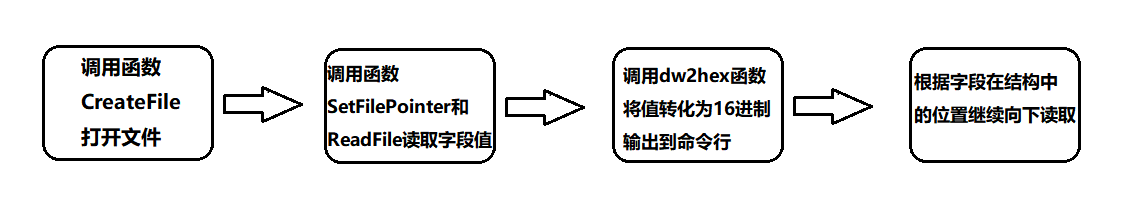
2.从文件的头部开始，调用SetFilePointer和ReadFile函数读取相应字段的值；

3.将读取出来的值通过dw2hex函数转化为16进制输出到命令行；

4.根据文件中其他字段与e\_magic和e\_lfanew的结构关系依次读取Signature、NumberOfSections、TimeDateStamp、Characteristics、AddressOfEntryPoint、ImageBase、SectionAlignment、FileAlignment的值并转化为16进制输出到命令行；

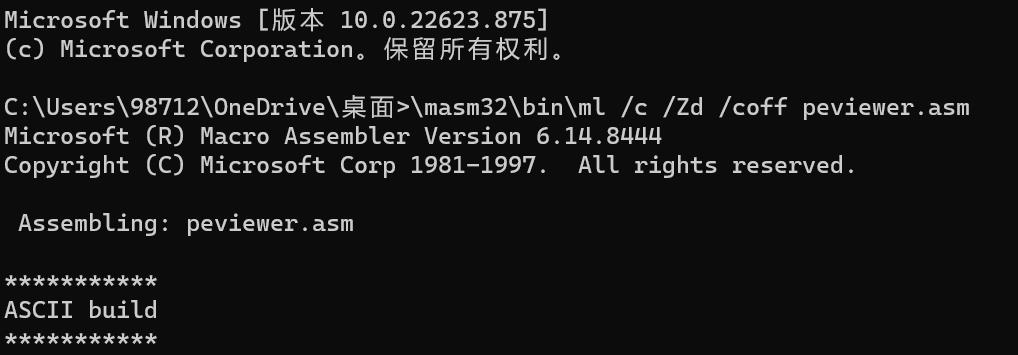
5.关闭。

* 控制流图：



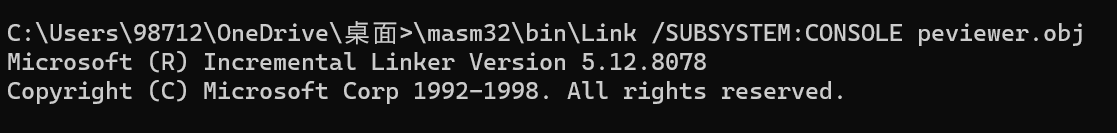
**2. 编译：使用ml将peviewer.asm文件汇编到peviewer.obj目标文件。**

编译命令：“\masm32\bin\ml /c /Zd /coff peviewer.asm”

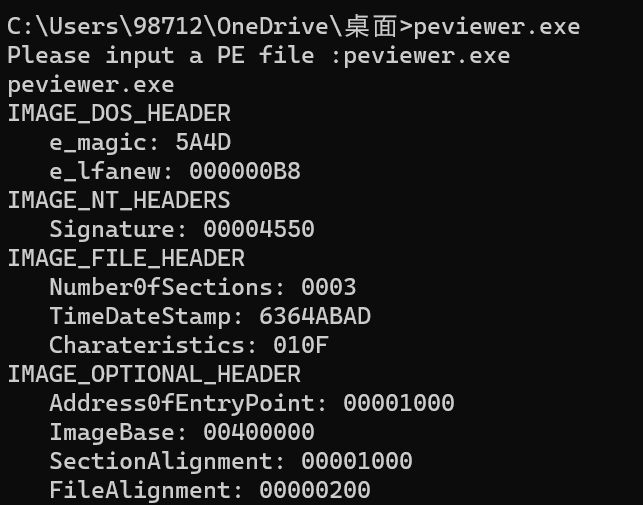


**3. 链接：使用link将目标文件peviewer.obj链接成peviewer.exe可执行文件。**

链接命令：“\masm32\bin\Link /SUBSYSTEM:CONSOLE peviewer.obj”



**4. 测试：直接执行peviewer.exe可执行文件。**

****

**五、源代码**

.386

.model flat, stdcall

option casemap :none

include \masm32\include\windows.inc

include \masm32\include\kernel32.inc

include \masm32\include\masm32.inc

includelib \masm32\lib\masm32.lib

includelib \masm32\lib\kernel32.lib

.data

    output db 100 DUP(0)     ;最后输出的字符串

    fileName db 100 DUP(0)   ;文件名

    hFile HANDLE 0

    content db 4000 DUP(0)

    e\_lfnew dd 0

    ;定义的待输出字符串

    str1 db "Please input a PE file :",0

    str2 db 0Ah,"IMAGE\_DOS\_HEADER",0Ah,"   e\_magic: ",0

    str3 db 0Ah,"   e\_lfanew: ",0

    str4 db 0Ah,"IMAGE\_NT\_HEADERS",0Ah,"   Signature: ",0

    str5 db 0Ah,"IMAGE\_FILE\_HEADER",0Ah,"   Number0fSections: ",0

    str6 db 0Ah,"   TimeDateStamp: ",0

    str7 db 0Ah,"   Charateristics: ",0

    str8 db 0Ah,"IMAGE\_OPTIONAL\_HEADER",0Ah,"   Address0fEntryPoint: ",0

    str9 db 0Ah,"   ImageBase: ",0

    str10 db 0Ah,"   SectionAlignment: ",0

    str11 db 0Ah,"   FileAlignment: ",0

.code

start:

    invoke StdOut, ADDR str1

    invoke StdIn, ADDR fileName, 100

    invoke StdOut, ADDR fileName

    ;调用函数CreateFile来打开文件

    invoke CreateFile, ADDR fileName,\

                       GENERIC\_READ,\

                       FILE\_SHARE\_READ,\

                       0,\

                       OPEN\_EXISTING,\

                       FILE\_ATTRIBUTE\_ARCHIVE,\

                       0

    ;调用函数SetFilePointer和ReadFile读取

    mov hFile, eax

    invoke SetFilePointer, hFile,\

                           0,\

                           0,\

                           FILE\_BEGIN

    invoke ReadFile, hFile,\

                     ADDR content,\

                     4000,\

                     0,\

                     0

    ;e\_magic转化为16进制输出

    mov eax, 0

    mov ax, WORD PTR content

    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str2

    invoke StdOut, ADDR output+4

    ;e\_lfanew转化为16进制输出

    mov eax, 0

    mov eax, DWORD PTR [content+3ch]

    mov e\_lfnew, eax

    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str3

    invoke StdOut, ADDR output

    ;Signature转化为16进制输出

    lea ebx, content

    add ebx, e\_lfnew

    mov eax, 0

    mov eax, DWORD PTR [ebx]

    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str4

    invoke StdOut, ADDR output

    ;Number0fSections转化为16进制输出

    lea ebx, content

    add ebx, e\_lfnew

    mov eax, 0

    mov ax, WORD PTR [ebx+6h]

    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str5

    invoke StdOut, ADDR output+4

    ;TimeDateStamp转化为16进制输出

    lea ebx, content

    add ebx, e\_lfnew

    mov eax, 0

    mov eax, DWORD PTR [ebx+8h]

    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str6

    invoke StdOut, ADDR output

    ;Charateristics转化为16进制输出

    lea ebx, content

    add ebx, e\_lfnew

    mov eax, 0

    mov ax, WORD PTR [ebx+16h]

    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str7

    invoke StdOut, ADDR output+4

    ;AddressOfEntryPoint转化为16进制输出

    lea ebx, content

    add ebx, e\_lfnew

    mov eax, 0

    mov eax, DWORD PTR [ebx+28h]

    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str8

    invoke StdOut, ADDR output

    ;ImageBase转化为16进制输出

    lea ebx, content

    add ebx, e\_lfnew

    mov eax, 0

    mov eax, DWORD PTR [ebx+34h]

    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str9

    invoke StdOut, ADDR output

    ;SectionAlignment转化为16进制输出

    lea ebx, content

    add ebx, e\_lfnew

    mov eax, 0

    mov eax, DWORD PTR [ebx+38h]

    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str10

    invoke StdOut, ADDR output

    ;FileAlignment转化为16进制输出

    lea ebx, content

    add ebx, e\_lfnew

    mov eax, 0

    mov eax, DWORD PTR [ebx+3Ch]

    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str11

    invoke StdOut, ADDR output

    ;调用函数CloseHandle关闭句柄

    invoke CloseHandle, hFile

    invoke ExitProcess, 0

END start