

**汇编语言与逆向技术实验报告**

**实验五：PEViewer**

****

学 院网络空间安全学院

专 业信息安全

学 号 2212046

姓 名 王昱

班 级 信息安全班

1. **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
2. **实验环境**

Windows操作系统，MASM32编译环境

1. **实验原理**

**（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。

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。

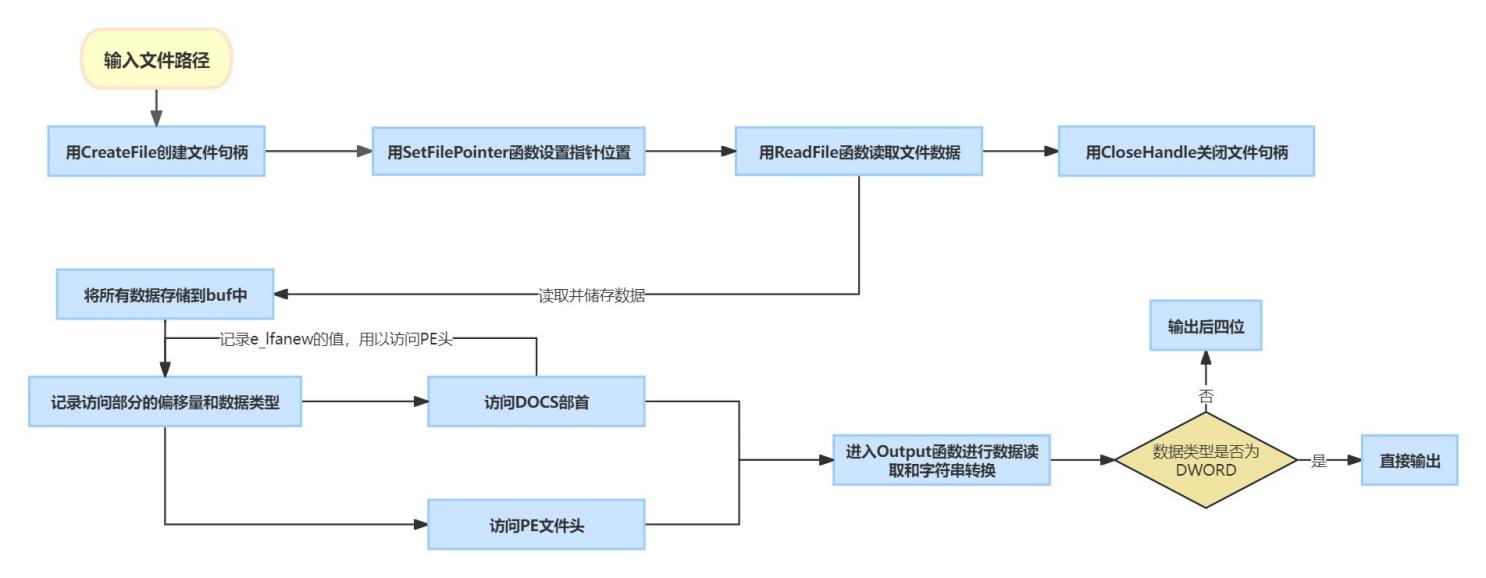
**（2） Windows文件读操作**

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

1. **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；
2. **实验过程**
3. **设计说明**

**在本次实验中，首先让用户输入一个 PE 文件的路径，然后调用 CreateFile 函数打开文件，返回文件句柄，并保存到 hfile 变量中。然后，将文件指针设置到文件开头，调用 ReadFile 函数读取文件内容到 buf 缓冲区中，用于后续的分析和输出。接着，定义一个 Output 过程，用于输出指定位置的数据。该过程的主要步骤是：将缓冲区的地址放入 esi 寄存器，然后加上偏移量 edx 和当前读取的位置 ini，得到要输出的数据的地址。将数据放入 eax 寄存器，并备份到 ebx 寄存器（用于记录 e\_lfanew 的值）。调用 dw2hex 函数，将数据转换为十六进制字符串，并保存到 buf1 缓冲区中。判断数据的长度，如果是 DWORD 类型的数据，就直接输出 buf1 中的内容；如果是 WORD 类型的数据，就只输出 buf1 中的后四位。输出换行符，结束该过程。然后，按照 PE 文件的格式，依次输出 DOS 头和 PE 头中的各个字段的值，每次输出前都输出相应的提示信息，例如 “IMAGE\_DOS\_HEADER”，“e\_magic” 等。每次输出时，都要设置好偏移量 edx，数据长度 ecx，当前读取的位置 ini（DOS 头为 0，PE 头为 e\_lfanew 的值），然后调用 Output 过程。最后，关闭文件句柄，退出程序。😊**

1. **控制流图**

****

1. **源代码**

**以下是实验的源代码和注释：**

**.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\kernel32.lib**

**includelib \masm32\lib\masm32.lib**

**.data**

**input BYTE "Please input a PE file:",0 ;提示用户输入一个 PE 文件的路径**

**hfile DWORD 0,0 ;存储打开的文件句柄**

**buf DWORD 4000 DUP(0) ;存储文件内容的缓冲区**

**ini DWORD 0 ;记录当前读取的位置是 DOS 头还是 PE 头**

**temp DWORD 0 ;临时变量，用于保存寄存器的值**

**buf1 DWORD 4000 DUP(0);存储转换为十六进制字符串的数据**

**string BYTE 100 DUP(0) ;存储用户输入的文件路径**

**endl BYTE 0Ah,0Dh,0;换行符**

**que1 BYTE "IMAGE\_DOS\_HEADER",0Ah,0Dh,0;以下是输出的提示信息**

**que2 BYTE " e\_magic:",0**

**que3 BYTE " e\_lfanew:",0**

**que4 BYTE "IMAGE\_DOS\_HEADER",0Ah,0Dh,0**

**que5 BYTE " Signature:",0**

**que6 BYTE "IMAGE\_DOS\_HEADER",0Ah,0Dh,0**

**que7 BYTE " NumberOfSections:",0**

**que8 BYTE " TimeDateStamp:",0**

**que9 BYTE " Charateristics:",0**

**que10 BYTE "IMAGE\_DOS\_HEADER",0Ah,0Dh,0**

**que11 BYTE " AddressOfEntryPoint:",0**

**que12 BYTE " ImageBase:",0**

**que13 BYTE " SectionAlignment:",0**

**que14 BYTE " FileAlignment:",0**

**.code**

**Output PROC ;输出指定位置的数据**

**mov esi,OFFSET buf;将缓冲区的地址放入 esi**

**add esi,edx;加上偏移量 edx**

**add esi,ini ;加上当前读取的位置 ini**

**mov eax,DWORD PTR[esi];将数据放入 eax**

**mov ebx,eax;将数据备份到 ebx**

**invoke dw2hex,eax,addr buf1 ;将数据转换为十六进制字符串**

**cmp ecx,8;判断是否是 DWORD 类型的数据**

**mov ecx,temp;恢复 ecx 的值**

**jne L1;如果不是，跳转到 L1**

**invoke StdOut,addr buf1;如果是，直接输出**

**jmp L2**

**L1:**

**invoke StdOut,addr [buf1+4];如果不是，只输出低位的两个字节**

**L2:**

**invoke StdOut,addr endl;输出换行符**

**ret**

**Output ENDP**

**start:**

**invoke StdOut,addr input;输出提示信息**

**invoke StdIn,addr string,100;获取用户输入的文件路径**

**invokeCreateFile,addrstring,GENERIC\_READ,FILE\_SHARE\_READ,0,OPEN\_EXISTING,FILE\_ATTRIBUTE\_ARCHIVE,0;打开文件，返回文件句柄**

**mov hfile,eax;将文件句柄保存到 hfile**

**invoke SetFilePointer,hfile,0,0,FILE\_BEGIN;将文件指针设置到文件开头**

**invoke ReadFile,hfile,addr buf,4000,0,0;读取文件内容到缓冲区**

**invoke StdOut,addr que1;输出提示信息**

**invoke StdOut,addr que2**

**mov edx,0;设置偏移量为 0**

**mov ini,edx;设置当前读取的位置为 DOS 头**

**mov ecx,4;设置输出的数据长度为 4**

**mov temp,ecx;将 ecx 的值保存到 temp**

**call Output;调用输出函数**

**invoke StdOut,addr que3;输出提示信息**

**mov edx,3ch;设置偏移量为 3ch**

**mov ecx,8;设置输出的数据长度为 8**

**mov temp,ecx;将 ecx 的值保存到 temp**

**call Output;调用输出函数**

**invoke StdOut,addr que4;输出提示信息**

**invoke StdOut,addr que5**

**mov ini,ebx;将 e\_lfanew 的值赋给 ini，设置当前读取的位置为 PE 头**

**mov edx,0;设置偏移量为 0**

**call Output;调用输出函数**

**invoke StdOut,addr que6;输出提示信息**

**invoke StdOut,addr que7**

**mov edx,6h;设置偏移量为 6h**

**mov ecx,4;设置输出的数据长度为 4**

**mov temp,ecx;将 ecx 的值保存到 temp**

**call Output;调用输出函数**

**invoke StdOut,addr que8;输出提示信息**

**mov edx,8h;设置偏移量为 8h**

**mov ecx,8;设置输出的数据长度为 8**

**mov temp,ecx;将 ecx 的值保存到 temp**

**call Output;调用输出函数**

**invoke StdOut,addr que9;输出提示信息**

**mov edx,16h;设置偏移量为 16h**

**mov ecx,4;设置输出的数据长度为 4**

**mov temp,ecx;将 ecx 的值保存到 temp**

**call Output;调用输出函数**

**invoke StdOut,addr que10;输出提示信息**

**invoke StdOut,addr que11**

**mov edx,28h;设置偏移量为 28h**

**mov ecx,8;设置输出的数据长度为 8**

**mov temp,ecx;将 ecx 的值保存到 temp**

**call Output;调用输出函数**

**invoke StdOut,addr que12;输出提示信息**

**mov edx,34h;设置偏移量为 34h**

**call Output;调用输出函数**

**invoke StdOut,addr que13;输出提示信息**

**mov edx,38h;设置偏移量为 38h**

**call Output;调用输出函数**

**invoke StdOut,addr que14;输出提示信息**

**mov edx,3ch;设置偏移量为 3ch**

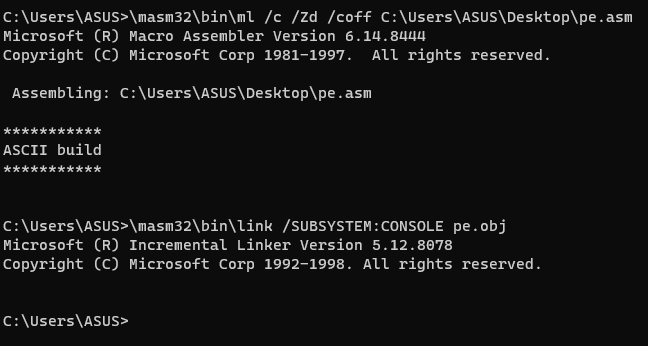
**call Output;调用输出函数**

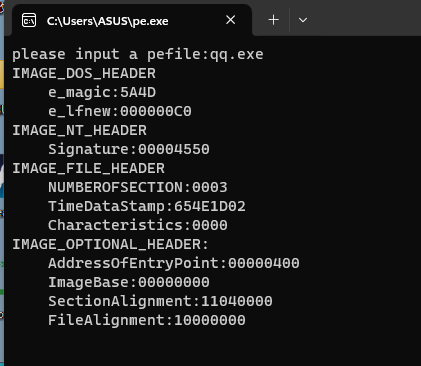
**invoke CloseHandle,hfile;关闭文件句柄**

**invoke ExitProcess,0**

**end start**

1. **测试截图**

****

****

**测试结果正确**

1. **实验结论及心得体会**

**通过这个实验，我学习了如何使用汇编语言编写一个简单的程序，用于分析和输出 PE 文件的格式和内容。我了解了 PE 文件的结构，包括 DOS 头，PE 头，节表，节等部分，以及它们的各个字段的含义和作用。**

不仅如此，本次实验加深了我对PE文件结构的认识，更加清晰的明白PE文件结构有哪些部分，每个部分存有哪些值。我**掌握了如何使用 MASM32 的库函数和自定义的过程，来打开，读取，转换，和显示文件的数据。我也发现了汇编语言的优点和缺点，例如它可以直接操作寄存器和内存，提高程序的效率，但也需要注意数据的类型，长度，地址，顺序等细节，避免出错。总的来说，这个实验对我提高了编程能力和理解计算机系统的原理有很大的帮助。**