

2021/02/02_PE_第2课_PE结构

笔记本: PE

创建时间: 2021/2/2 星期二 10:48

作者: ileemi

- [PE 结构](#)
 - [IMAGE_DOS_HEADER](#)
 - [IMAGE_NT_HEADERS](#)
 - [IMAGE_FILE_HEADER](#)
 - [IMAGE_OPTIONAL_HEADER32](#)

PE 结构

结构如下:

```
IMAGE_DOS_HEADER  // 64个字节
DOS_STUB          // 一般为176个字节
IMAGE_NT_HEADERS  // IMAGE_NT_HEADERS32  -- 248个字节
    -- IMAGE_FILE_HEADER  // 文件头
    -- IMAGE_OPTIONAL_HEADER  // 选项头
    -- IMAGE_DATA_DIRECTORY  // 数据目录
    ...
IMAGE_SECTION_HEADER  // 40个字节
...
IMAGE_SECTION_HEADER
SECTION_DATA
...
SECTION_DATA
// 附加数据, 用户可以进行自定义
// 受对齐值的影响, 在可执行文件最后追加数据, 有可能被映射到内存中
user data
```

IMAGE_DOS_HEADER

结构如下:

```
typedef struct _IMAGE_DOS_HEADER { // DOS .EXE header
    WORD    e_magic; // Magic number
    WORD    e_cblp; // Bytes on last page of file
```

```

WORD    e_cp; // Pages in file
WORD    e_crlc; // Relocations
WORD    e_cparhdr; // Size of header in paragraphs
WORD    e_minalloc; // Minimum extra paragraphs needed
WORD    e_maxalloc; // Maximum extra paragraphs needed
WORD    e_ss; // Initial (relative) SS value
WORD    e_sp; // Initial SP value
WORD    e_csum; // Checksum
WORD    e_ip; // Initial IP value
WORD    e_cs; // Initial (relative) CS value
WORD    e_lfarlc; // File address of relocation table
WORD    e_ovno; // Overlay number
WORD    e_res[4]; // Reserved words
WORD    e_oemid; // OEM identifier (for e_oeminfo)
WORD    e_oeminfo; // OEM information; e_oemid specific
WORD    e_res2[10]; // Reserved words
LONG    e_lfanew; // File address of new exe header
} IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;

```

DOS(MZ标志): 在 "winnt.h" 头文件中, 每种文件都有自己的文件格式以及固定的文件头。文件格式由软件本身自己定义。

注意: 16位操作系统中一个分页为512字节, 32位程序中保留DOS文件头是为了兼容16位程序。通过64位操作系统编译生成的32位可执行程序在16位操作系统中程序是不能正常运行的, 会提示 "This program cannot be run in DOS mode." 字符串。32位程序在32位操作系统上可以将 "DOS头代码 (除 "e_magic" 两个字节 和 "e_lfanew" 四个字节之外) " 以及 "DOS_TUB" 代码都可以将其二进制代码全部替换, 不会影响程序的运行结果。32位、64位文件格式没有区别。

DOS_TUB: DOS的代理代码 (DOS头 (64字节) 之后的代码), 一般为176个字节。

"DOS头" 结构体一共64个字节, 其中第一个字段 "**e_magic**" 和最后一个字段 "**e_lfanew**" 最为重要, 作用如下:

- **e_magic:** DOS头标志位, 其值恒为0x5A4D ("MZ"), 操作系统只定义可执行文件的格式, 并根据文件头判断其是否是可执行文件格式。
- **e_lfanew:** 执行新的文件格式, 记录其偏移地址, 表示NT头部在文件中的偏移。

IMAGE_NT_HEADERS

结构如下:

```

typedef struct _IMAGE_NT_HEADERS {
    DWORD Signature;
    IMAGE_FILE_HEADER FileHeader;

```

```
    IMAGE_OPTIONAL_HEADER32 OptionalHeader;
} IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;
```

NT头，Windows操作系统也称为 "NT" 操作系统。新操作系统的文件格式。一般默认为 248 个字节。

区分32位、64位（只会影响结构体的大小，不会影响文件格式）：

- IMAGE_NT_HEADERS
- IMAGE_NT_HEADERS64

IMAGE_NT_HEADERS 结构体中的三个成员都重要，第一个成员为标志（PE），后两个为嵌套的结构体，作用如下：

- DWORD Signature: NT头为固定的四个字节，类似DOS头的 "e_magic" 字段，其值为 "0x00004550"。修改任意字节，操作系统就无法解析其是一个有效的可执行程序。
- IMAGE_FILE_HEADER FileHeader（文件头）：存储着PE文件的基本信息。
- IMAGE_OPTIONAL_HEADER32 OptionalHeader（选项头）：存储着PE文件加载的信息。

IMAGE_FILE_HEADER

文件头，存储着PE文件的基本信息。

结构如下：

```
typedef struct _IMAGE_FILE_HEADER {
    // Architecture type of the computer
    WORD Machine;
    // The number of sections
    WORD NumberOfSections;
    // The low 32 bits of the time stamp of the image
    DWORD TimeDateStamp;
    // The offser of the symbol table
    DWORD PointerToSymbolTable;
    // The number of symbols in the symbol table
    DWORD NumberOfSymbols;
    // The size of the optional header
    WORD SizeOfOptionalHeader;
    // The characteristics of the image
    WORD Characteristics;
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

成员作用如下：

- Machine: 标识运行平台
- NumberOfSections: 节区的数量

- TimeDateStamp: 时间戳, 记录文件创建的时间, 可参考
- PointerToSymbolTable: 符号表地址
- NumberOfSymbols: 符号表的数量
- **SizeOfOptionalHeader**: 选项头结构的大小, 大小不固定 (变长)
- Characteristics: 该可执行文件的特征属性 (可执行, dll文件等)

Characteristics

Specifies the characteristics of the image.

Value	Meaning
IMAGE_FILE_RELOCS_STRIPPED	Relocation information is stripped from the file.
IMAGE_FILE_EXECUTABLE_IMAGE	The file is executable (there are no unresolved external references).
IMAGE_FILE_LINE_NUMS_STRIPPED	Line numbers are stripped from the file.
IMAGE_FILE_LOCAL_SYMS_STRIPPED	Local symbols are stripped from file.
IMAGE_FILE_AGGRESSIVE_WS_TRIM	Aggressively trim the working set.
IMAGE_FILE_LARGE_ADDRESS_AWARE	The application can handle addresses larger than 2 GB.
IMAGE_FILE_BYTES_REVERSED_LO	Bytes of the word are reversed.
IMAGE_FILE_32BIT_MACHINE	Computer supports 32-bit words.
IMAGE_FILE_DEBUG_STRIPPED	Debugging information is stored separately in a .dbg file.
IMAGE_FILE_REMOVABLE_RUN_FROM_SWAP	If the image is on removable media, copy and run from the swap file.
IMAGE_FILE_NET_RUN_FROM_SWAP	If the image is on the network, copy and run from the swap file.
IMAGE_FILE_SYSTEM	System file.
IMAGE_FILE_DLL	DLL file.
IMAGE_FILE_UP_SYSTEM_ONLY	File should be run only on a uniprocessor computer.
IMAGE_FILE_BYTES_REVERSED_HI	Bytes of word are reversed.

IMAGE_OPTIONAL_HEADER32

IMAGE_OPTIONAL_HEADER32 (选项头) : 存放各种API使用的情况。

结构如下:

```
typedef struct _IMAGE_OPTIONAL_HEADER {
    //
    // Standard fields.
    //
    WORD Magic;
    BYTE MajorLinkerVersion;
    BYTE MinorLinkerVersion;
    DWORD SizeOfCode;
    DWORD SizeOfInitializedData;
    DWORD SizeOfUninitializedData;
    DWORD AddressOfEntryPoint;
    DWORD BaseOfCode;
    DWORD BaseOfData;
    //
    // NT additional fields.
    //
    DWORD ImageBase;
    DWORD SectionAlignment;
    DWORD FileAlignment;
```

```

WORD MajorOperatingSystemVersion;
WORD MinorOperatingSystemVersion;
WORD MajorImageVersion;
WORD MinorImageVersion;
WORD MajorSubsystemVersion;
WORD MinorSubsystemVersion;
DWORD Win32VersionValue;
DWORD SizeOfImage;
DWORD SizeOfHeaders;
DWORD CheckSum;
WORD Subsystem;
WORD DllCharacteristics;
DWORD SizeOfStackReserve;
DWORD SizeOfStackCommit;
DWORD SizeOfHeapReserve;
DWORD SizeOfHeapCommit;
DWORD LoaderFlags;
DWORD NumberOfRvaAndSizes;
IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];

} IMAGE_OPTIONAL_HEADER32, *PIMAGE_OPTIONAL_HEADER32;

```

各成员作用：

- **Magic**：文件状态（16位、32位）
 - **IMAGE_NT_OPTIONAL_HDR32_MAGIC** (0x10b)
 - **IMAGE_NT_OPTIONAL_HDR64_MAGIC** (0x20b)
 - **IMAGE_ROM_OPTIONAL_HDR32_MAGIC** (0x107)
- **MajorLinkerVersion**：链接器的版本号（主）
- **MinorLinkerVersion**：链接器的版本号（次），供参考程序的编译平台。
- **SizeofCode**：代码区段的总大小（汇编代码的总字节数），说明性字段（参考），**修改该成员的数据（AAAAAAA）可用于反调试器（ollydbg调试器会读取这个字段并申请对应的空间用于反汇编）。调试器对程序进行反汇编时不应该读取该字段数据。**
- **SizeOfInitializedData**：已初始化数据区的总大小，说明性字段
- **SizeOfUninitializedData**：未初始化数据区的总大小，在磁盘中不占用空间，加载到内存后会预留空间，一般存储在.bss段中，说明性字段
- **AddressOfEntryPoint**：指向入口点函数的指针，**相对于主模块的基地址**（也称为EP（Relative Virtual Address），OEP为原始入口点）。**为了防止随机基址，值一般不为绝对地址，一般为相对主模块地址的偏移值。**对于可执行文件这是起始地址，对于设备文件这是初始化函数地址，对于DLL入口点函数指针是可选的。当没有入口点函数指针时，这个成员的值为0，PE下数两行半。
- **BaseOfCode**：指向代码区段开始的指针，相对于模块的基地址，说明性字段
- **BaseOfData**：指向数据区段开始的指针，相对于模块的基地址，说明性字段
- **ImageBase**：载入到内存时该image文件第一个字节的地址即基地址（主模块的基址），这个值是64K bytes的整数倍。DLL默认为0x10000000，应用程序默

认为0x00400000，除了在Windows CE它是0x00010000，建议装载地址

- **SectionAlignment**: 内存对齐，映射到内存中段的对齐方式，以页为单位。该值必须大于或等于FileAlignment成员。默认值为系统的页面大小。
- **FileAlignment**: 文件对齐，在磁盘中的段的对齐方式，以字节为单位。该值应为512到64K（含）之间的2的幂。默认值为512。如果SectionAlignment成员小于系统页面大小，则此成员必须与SectionAlignment 相同
- **MajorSubsystemVersion**: 子系统版本，这个值是必须的，不足512字节，补到512字节。提高内存读取效率。
- **MajorOperatingSystemVersion**: 操作系统版本（主），参考值
- **MinorOperatingSystemVersion**: 操作系统版本（次），参考值
- **MajorImageVersion**: 镜像版本（主），参考值
- **MinorImageVersion**: 镜像版本（次），参考值
- **MajorSubsystemVersion**: 子系统版本（主）
- **MinorSubsystemVersion**: 子系统版本（次）
- **Win32VersionValue**: 版本值，参考值
- **SizeOfImage**: 镜像占用的内存大小，载入内存后image的大小（进行了内存对齐之后），由操作系统的依赖，必须对齐，必须不多不少
- **SizeOfHeaders**: 头的总字节大小，以下结构的总字节大小（按文件对齐后）：
 - e_lfanew member of IMAGE_DOS_HEADER4 byte signature
 - size of IMAGE_FILE_HEADER
 - size of optional header
 - size of all section headers
- **Checksum**: 对整个文件进行校验和，由编译器计算，操作系统进行验证，防止修改文件中的数据，**操作系统只校验驱动程序**，一般不校验可执行程序（R3不检查）。根据算法可对齐进行修复。
- **Subsystem**: 子系统（可执行程序，"1" -- 驱动程序（不能双击运行），"2" -- 窗口程序，"3" -- 控制台）
- **DllCharacteristics**: 随机基址（为了提高软件的安全性，在可执行文件中是一个标志位，PE下数5行半，第三位改为4开启随机基址(前提需要修改可执行文件中的代码))，Windows上可执行程序的随机基址其实是伪随机（可执行文件中保存了一个重定位表），有标志位。
- **SizeOfStackReserve**: 初始化时栈的大小（预留），0x10000，不建议修改，链接器有提供对应的参数选项。
- **SizeOfStackCommit**: 初始化时实际提交（软件运行起来需要先使用的栈空间）的栈的大小，0x1000，不建议修改
- **SizeOfHeapReserve**: 初始化时保留的堆的大小（预留），0x10000，不建议修改，向操作系统申请空间（汇编中可使用 VirtualAlloc 申请空间）。
- **SizeOfHeapCommit**: 初始化时实际提交（软件运行起来需要先使用的堆空间）的堆的大小，0x1000，不建议修改，**编译时，链接器根据提交的大小去申请对应的空间。**
- **LoaderFlags**: 调试器相关的成员，实用性不大
- **NumberOfRvaAndSizes**: 记录数据目录表中的数量
- **DataDirectory**: 数据目录 IMAGE_DATA_DIRECTORY 结构的数组（动态数组），每项8各字节，表的类型由数组下标进行决定。

