

3、PE 文件结构

① MS-DOS 头

MS-DOS 头在 `winnt.h` 中定义成为 `IMAGE_DOS_HEADER`，这个结构中，最需要关心的是成员 `e_lfanew`，它给出了 PE Header 在文件中的偏移量。比如，`e_lfanew` 的值是 `0xE0`，则 PE Header 在文件距离开头 `0xE0` 处。

② MS DOS 2.0 Stub Program

这是一段 DOS 程序，如果把 win32 的可执行文件放到 dos 上执行，这段代码将在屏幕上显示类似于“This program can not run in dos”的信息。

③ Magic Number 和 PE Header

这才是真正的 win32 可执行文件的开始。Magic Number 是一个 DWORD 类型的数，值是 `0x4550`，对应的 ASCII 值就是“PE”。PE Header 在 `winnt.h` 里被定义成为：

```
typedef struct _IMAGE_FILE_HEADER {  
    WORD    Machine;  
    WORD    NumberOfSections;  
    DWORD   TimeDateStamp;  
    DWORD   PointerToSymbolTable;  
    DWORD   NumberOfSymbols;  
    WORD    SizeOfOptionalHeader;  
    WORD    Characteristics;  
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

其中：

Machine: 声明 PE 文件是在那种 CPU 架构下运行，它可以是下表中所列出的值：（通常 PE 文件运行在 x86 系列的 CPU 架构下，这个值就应该是 `0x14C`。）

标志	值	说明
IMAGE_FILE_MACHINE_UNKNOWN	0x0	假定在所有机器上运行
IMAGE_FILE_MACHINE_AM33	0x1d3	Matsushita AM33
IMAGE_FILE_MACHINE_AMD64	0x8664	x64
IMAGE_FILE_MACHINE_ARM	0x1c0	ARM little endian
IMAGE_FILE_MACHINE_EBC	0xebc	EFI byte code
IMAGE_FILE_MACHINE_I386	0x14c	Intel 386 家族及其兼容 CPU
IMAGE_FILE_MACHINE_IA64	0x200	Intel Itanium 处理器家族
IMAGE_FILE_MACHINE_M32R	0x9041	Mitsubishi M32R little endian
IMAGE_FILE_MACHINE_MIPS16	0x266	MIPS16
IMAGE_FILE_MACHINE_MIPSFPU	0x366	MIPS with FPU
IMAGE_FILE_MACHINE_MIPSFPU16	0x466	MIPS16 with FPU
IMAGE_FILE_MACHINE_POWERPC	0x1f0	Power PC little endian
IMAGE_FILE_MACHINE_POWERPCFP	0x1f1	有浮点支持的 Power PC
IMAGE_FILE_MACHINE_R4000	0x166	MIPS little endian
IMAGE_FILE_MACHINE_SH3	0x1a2	Hitachi SH3
IMAGE_FILE_MACHINE_SH3DSP	0x1a3	Hitachi SH3 DSP
IMAGE_FILE_MACHINE_SH4	0x1a6	Hitachi SH4

标志	值	说明
IMAGE_FILE_MACHINE_SH5	0x1a8	Hitachi SH5
IMAGE_FILE_MACHINE_THUMB	0x1c2	Thumb
IMAGE_FILE_MACHINE_WCEMIPSV2	0x169	MIPS little-endian WCE v2

NumberOfSections: 表示 PE 文件中节的数量。

TimeDateStamp: 链接器产生这个文件的时间,是自从 1969 年 12 月 31 日 4:00 P.M. 之后的总秒数。

PointerToSymbolTable、NumberOfSymbols: 一般只对 COFF (Common Object File Format) 格式文件有用。

SizeOfOptionalHeader: 它是 Optional header 的大小,也就是 sizeof(IMAGE_OPTIONAL_HEADER) (Optional header 被定义成为 IMAGE_OPTIONAL_HEADER 结构)。

Characteristics: 声明这个 PE 文件的性质,它可以是下表列出的值,并且可以按位或:(对于一个 DLL 文件,这个值应该通常是 0x2102; 对于一个 EXE 文件,这个值通常是 0x0103)

标志	值	描述
IMAGE_FILE_RELOCS_STRIPPED	0x0001	用于 Windows CE, Windows NT 以及后续操作系统。声明此 PE 文件没有基础重定位,并且必须装载到预先定义的基地址。如果基地址不可用,装载器将报错。
IMAGE_FILE_EXECUTABLE_IMAGE	0x0002	这是一个可执行文件。
IMAGE_FILE_LINE_NUMS_STRIPPED	0x0004	这一位应该置零。
IMAGE_FILE_LOCAL_SYMS_STRIPPED	0x0008	这一位应该置零。
IMAGE_FILE_AGGRESSIVE_WS_TRIM	0x0010	建议不要再 Windows 2000 机器后续系统中使用,应该置零。
IMAGE_FILE_LARGE_ADDRESS_AWARE	0x0020	程序可以处理大于 2G 的地址。
	0x0040	此位为将来使用而保留。
IMAGE_FILE_BYTES_REVERSED_LO	0x0080	这一位应该置零。
IMAGE_FILE_32BIT_MACHINE	0x0100	机器以 32 位架构为基础。
IMAGE_FILE_DEBUG_STRIPPED	0x0200	调试信息已经被移除。
IMAGE_FILE_REMOVABLE_RUN_FROM_SWAP	0x0400	如果文件位于可移动的媒体上,那么将整个文件读入交换文件。
IMAGE_FILE_NET_RUN_FROM_SWAP	0x0800	如果文件位于网络上,那么将整个文件读入交换文件。
IMAGE_FILE_SYSTEM	0x1000	文件是一个系统文件,不是用户文件
IMAGE_FILE_DLL	0x2000	文件是一个动态链接库,虽然它们不能直接运行,也被认为是一个可执行文件。

标志	值	描述
IMAGE_FILE_UP_SYSTEM_ONLY	0x4000	只能在单芯片机器上运行
IMAGE_FILE_BYTES_REVERSED_HI	0x8000	这一位应该置零。

① Optional header

从字面上看，这个文件头是可选的，但实际上它是 PE 文件中必不可少的。它在 winnt.h 中被定义称为：

```
typedef struct _IMAGE_OPTIONAL_HEADER {
    WORD    Magic;
    BYTE    MajorLinkerVersion;
    BYTE    MinorLinkerVersion;
    DWORD   SizeOfCode;
    DWORD   SizeOfInitializedData;
    DWORD   SizeOfUninitializedData;
    DWORD   AddressOfEntryPoint;
    DWORD   BaseOfCode;
    DWORD   BaseOfData;
    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;
```

其中：

Magic: 声明 PE 文件的状态，如果是普通的 PE 文件，值为 0x10B；如果是只读的，值为 0x107。

MajorLinkerVersion、MinorLinkerVersion: 链接器的版本号,但实际上这两个值并不可靠,某些链接器不会设置这两个值。

SizeOfCode: 所有代码节的大小。

SizeOfInitializedData、SizeOfUninitializedData: 所有已初始化数据节、未初始化数据节的大小。

AddressOfEntryPoint: 这是一个 RVA。当 PE 文件被装载到内存中以后,第一条可执行指令的地址。对于设备驱动,这是初始化函数的地址;对于 DLL,这个入口点是可选的,如果 DLL 没有入口点,则这个值必须为零。

BaseOfCode、BaseOfData: 代码的 RVA,已初始化数据的 RVA。

ImageBase: PE 文件的优先装载地址。比如,这个值是 0x10000000,则装载器会将 PE 文件优先装载到虚拟内存地址 0x10000000 中。通常 EXE 文件的这个值是 0x00400000;.DLL 文件的是 0x10000000。

SectionAlignment: PE 文件装载到内存后,节的对齐粒度,必须大于等于 FileAlignment。WIN32 下,一般是 0x1000,WIN64 下,一般是 0x2000。

FileAlignment: PE 文件中,节的对齐粒度。一般情况下是 0x200 的倍数。

MajorOperatingSystemVersion、MinorOperatingSystemVersion: 期望的操作系统版本。

MajorImageVersion、MinorImageVersion: 期望的 PE 文件版本,某些链接器不设定这个值。

MajorSubsystemVersion、MinorSubsystemVersion: 期望的子系统版本。

Win32VersionValue: 这个是保留的,必须是 0。

SizeOfImage: PE 文件装载到内存后,整个镜像的大小,必须是 SectionAlignment 的整数倍。

SizeOfHeaders: MS-DOS 头、MS DOS 2.0 Stub Program、Magic Number、PE Header 和 Optional header 大小之和,必须是 FileAlignment 的整数倍。

Checksum: 对于普通的 PE 文件,这个值是 0。

Subsystem: 声明 PE 文件在什么样的系统上运行,可以是下表中的值:(对于 WINDOWS 开发,通常选择第三项或者第四项)

Constant	Value	Description
IMAGE_SUBSYSTEM_UNKNOWN	0	未知子系统。
IMAGE_SUBSYSTEM_NATIVE	1	设备驱动以及 WINDOWS 内部程序。
IMAGE_SUBSYSTEM_WINDOWS_GUI	2	WINDOWS GUI 程序。
IMAGE_SUBSYSTEM_WINDOWS_CUI	3	WINDOWS 控制台程序。
IMAGE_SUBSYSTEM_POSIX_CUI	7	Posix 字符子系统程序。
IMAGE_SUBSYSTEM_WINDOWS_CE_GUI	9	Windows CE。
IMAGE_SUBSYSTEM_EFI_APPLICATION	10	可扩展固件程序。
IMAGE_SUBSYSTEM_EFI_BOOT_SERVICE_DRIVER	11	启动服务的 EFI 驱动。
IMAGE_SUBSYSTEM_EFI_RUNTIME_DRIVER	12	运行时的 EFI 驱动。
IMAGE_SUBSYSTEM_EFI_ROM	13	EFI 只读镜像。
IMAGE_SUBSYSTEM_XBOX	14	XBOX。

DllCharacteristics: 声明 DLL 文件的性质，可以是下表中的值：（如果没有特别需要，这个值是零）

常量	值	描述
	0x0001	保留的，必须为零。
	0x0002	保留的，必须为零。
	0x0004	保留的，必须为零。
	0x0008	保留的，必须为零。
IMAGE_DLL_CHARACTERISTICS_DYNAMIC_BASE	0x0040	DLL 可以在运行时被重置。
IMAGE_DLL_CHARACTERISTICS_FORCE_INTEGRITY	0x0080	强制进行代码完整性检查。
IMAGE_DLL_CHARACTERISTICS_NX_COMPAT	0x0100	映像是 NX 兼容的。
IMAGE_DLLCHARACTERISTICS_NO_ISOLATION	0x0200	不隔离映像文件。
IMAGE_DLLCHARACTERISTICS_NO_SEH	0x0400	不使用结构化异常处理。
IMAGE_DLLCHARACTERISTICS_NO_BIND	0x0800	不绑定映像
	0x1000	保留的，必须为零。
IMAGE_DLLCHARACTERISTICS_WDM_DRIVER	0x2000	WDM 驱动。
IMAGE_DLLCHARACTERISTICS_TERMINAL_SERVER_AWARE	0x8000	终端服务器

SizeOfStackReserve: 预留栈的大小，一般默认为 0x10000。

SizeOfStackCommit: 提交栈的大小，一般默认为 0x1000。

SizeOfHeapReserve: 预留堆的大小，一般默认为 0x10000。

SizeOfHeapCommit: 提交堆的大小，一般默认为 0x1000。

LoaderFlags: 保留的，必须为零。

NumberOfRvaAndSizes: 之后的数据目录的数量，强烈建议使用默认的 16。

DataDirectory: 数据目录，每一个对应一个节，声明该节的 RVA 和大小。

② Section headers

每一个节对应一个与之相关的节头，节头声明了节的大小、RVA 以及的性，在 winnt.h 中，节头定义如下：

```
#define IMAGE_SIZEOF_SHORT_NAME 8
typedef struct _IMAGE_SECTION_HEADER {
    BYTE    Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
        DWORD    PhysicalAddress;
        DWORD    VirtualSize;
    } Misc;
    DWORD    VirtualAddress;
```

```

        DWORD   SizeOfRawData;
        DWORD   PointerToRawData;
        DWORD   PointerToRelocations;
        DWORD   PointerToLinenumbers;
        WORD    NumberOfRelocations;
        WORD    NumberOfLinenumbers;
        DWORD   Characteristics;
    } IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;

```

其中：

Name: 一个 8 字节长度的变量，给出节的名字。如果名字中的字符少于 8 字节，则用 NULL 填充；如果刚好等于 8 字节，则不需要以 NULL 结束。一般情况下，代码节名称为“.text”，数据节为“.data”。

Misc: 这是一个联合体，在可执行映像中，使用的是 VirtualSize，声明对应的节的大小。

VirtualAddress: 当 PE 文件读入内存后，对应节的 RVA。

SizeOfRawData: 磁盘上，节根据 FileAlignment 对齐后的大小，必须是 FileAlignment 的倍数。

PointerToRawData: 从文件开头到对应节的偏移量。

PointerToRelocations

PointerToLinenumbers

NumberOfRelocations

NumberOfLinenumbers: 以上四个变量在可执行文件中用不到。

Characteristics: 声明节的性质：可以是下表中的值，并可以按位或：

标志	值	描述
IMAGE_SCN_CNT_CODE	0x00000020	节包含可执行代码。
IMAGE_SCN_CNT_INITIALIZED_DATA	0x00000040	节包含已经初始化的数据。
IMAGE_SCN_CNT_UNINITIALIZED_DATA	0x00000080	节包含未初始化的数据。
IMAGE_SCN_LNK_INFO	0x00000200	节包含注释或其他信息。只用于目标文件。
IMAGE_SCN_LNK_REMOVE	0x00000800	节不是映像的一部分，只用于目标文件。
IMAGE_SCN_LNK_COMDAT	0x00001000	节包含 COMDAT 数据。只用于目标文件。
IMAGE_SCN_GPREL	0x00008000	节包含引用全局指针的数据。
IMAGE_SCN_LNK_NRELOC_OVFL	0x01000000	节包含扩展重定位。
IMAGE_SCN_MEM_DISCARDABLE	0x02000000	根据需要，节可被废弃。
IMAGE_SCN_MEM_NOT_CACHED	0x04000000	节不可被缓存。
IMAGE_SCN_MEM_NOT_PAGED	0x08000000	节不可被分页。
IMAGE_SCN_MEM_SHARED	0x10000000	节可在内存中被共享。
IMAGE_SCN_MEM_EXECUTE	0x20000000	节可以执行。

标志	值	描述
IMAGE_SCN_MEM_READ	0x40000000	节可被读取。
IMAGE_SCN_MEM_WRITE	0x80000000	节可被写入。