PE文件 — PE头 — e\_magic，DOS签名，必须为MZ

— e\_lfanew，NT头偏移

— ……

— DOS存根

— NT头 — 签名结构体，PE文件格式核心标识“PE\0\0”

— 文件头 — Machine编码

— NumberOfSections，文件节区数量

— SizeOfOptionalHeader，特定架构下的可选头长度

— Characteristics，标识文件属性

— TimeDateStamp，纪录编译创建时间

— ……

— 可选头 — magic

— AdressOfEntryPoint，指出程序最先执行的代码起始地址

— ImageBase，指出文件优先装载地址

— SectionAlignment，节区内存中的最小单位

— FileAlignment，节区在磁盘中的最小单位

— SizeOfImage，指定PEImage在虚拟内存中所占的空间大小

— SubSystem，区分系统驱动文件和普通可执行文件

— NumberOfRvaAndSize，指定DataDirectory数组个数

— DataDirectory数组

— ……

— 节区头 — VirtualSize，内存中节区大小

— VirtualAddress，内存节区起始地址，由可选头SectionAlignment确定

— SizeOfRawData，磁盘中节区所占大小

— PointerToRawData，磁盘节区起始地址，由FileAlignment确定

— Characteristics，节区属性

— ……

（code、data、resource存在不同节区，节区头就由上面的结构体组成数组，每个结构体对应一个区）

DOS头

共64字节。

typedef struct \_IMAGE\_DOS\_HEADER {

WORD e\_magic; *// DOS签名 "MZ" (0x5A4D)，重要*

WORD e\_cblp; *// 字节数（最后页）*

WORD e\_cp; *// 页数*

WORD e\_crlc; *// 重定位项数*

WORD e\_cparhdr; *// 头部段数*

WORD e\_minalloc; *// 最小内存分配*

WORD e\_maxalloc; *// 最大内存分配*

WORD e\_ss; *// 初始SS值*

WORD e\_sp; *// 初始SP值*

WORD e\_csum; *// 校验和*

WORD e\_ip; *// 初始IP值*

WORD e\_cs; *// 初始CS值*

WORD e\_lfarlc; *// 重定位表偏移*

WORD e\_ovno; *// 覆盖号*

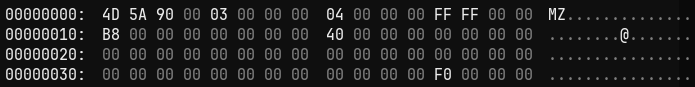
WORD e\_res[4]; *// 保留字段*

WORD e\_oemid; *// OEM标识符*

WORD e\_oeminfo; *// OEM信息*

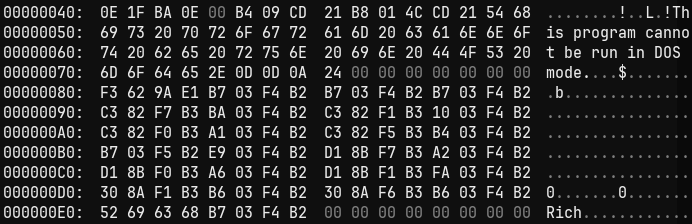
WORD e\_res2[10]; *// 保留字段*

LONG e\_lfanew; *// NT头偏移（PE文件起始位置），重要*

} IMAGE\_DOS\_HEADER, \*PIMAGE\_DOS\_HEADER;

DOS存根

在DOS系统下提示一句话，包含数据和代码，可以修改，但是修改不可覆盖DOS头和NT头，长度不能随意修改，如果缩短或增长需要修改地址，非常麻烦。



NT头

没什么说的。

typedef struct \_IMAGE\_NT\_HEADERS {

DWORD Signature; *// PE签名 "PE\0\0" (0x00004550)*

IMAGE\_FILE\_HEADER FileHeader; *// 文件头*

IMAGE\_OPTIONAL\_HEADER OptionalHeader; *// 可选头（32/64位）*

} IMAGE\_NT\_HEADERS32, \*PIMAGE\_NT\_HEADERS32;

*// 64位版本（IMAGE\_OPTIONAL\_HEADER64）*

typedef struct \_IMAGE\_NT\_HEADERS64 {

DWORD Signature;

IMAGE\_FILE\_HEADER FileHeader;

IMAGE\_OPTIONAL\_HEADER64 OptionalHeader;

} IMAGE\_NT\_HEADERS64, \*PIMAGE\_NT\_HEADERS64;

NT头：签名结构体

PE文件核心标识，内容“PE\0\0”



NT头：文件头

固定20字节。

typedef struct \_IMAGE\_FILE\_HEADER {

WORD Machine; *// 目标CPU架构（如0x014C=Intel 386）*

WORD NumberOfSections; *// 节区数量*

DWORD TimeDateStamp; *// 编译时间戳*

DWORD PointerToSymbolTable; *// 符号表偏移（调试用）*

DWORD NumberOfSymbols; *// 符号数量*

WORD SizeOfOptionalHeader; *// 可选头大小*

WORD Characteristics; *// 文件属性（如可执行/DLL）*

} IMAGE\_FILE\_HEADER, \*PIMAGE\_FILE\_HEADER;



NT头：可选头

长度由文件头里的SizeOfOptionalHeader确定，32位PE文件通常为0xE0（224字节），64位PE文件通常为0xF0（240字节）。

typedef struct \_IMAGE\_OPTIONAL\_HEADER32 {

*// 标准字段（所有PE文件）*

WORD Magic; *// 标识：0x10B=32位，0x20B=64位*

BYTE MajorLinkerVersion; *// 链接器主版本号*

BYTE MinorLinkerVersion; *// 链接器次版本号*

DWORD SizeOfCode; *// 所有代码段的总大小*

DWORD SizeOfInitializedData; *// 已初始化数据的总大小*

DWORD SizeOfUninitializedData; *// 未初始化数据（BSS）的总大小*

DWORD AddressOfEntryPoint; *// 入口点RVA（相对于ImageBase）*

DWORD BaseOfCode; *// 代码段的起始RVA*

DWORD BaseOfData; *// 数据段的起始RVA（仅32位存在）*

*// NT扩展字段（Windows专用）*

DWORD ImageBase; *// 进程内存中的优先加载地址*

DWORD SectionAlignment; *// 内存中的节区对齐粒度（通常0x1000）*

DWORD FileAlignment; *// 文件中的节区对齐粒度（通常0x200）*

WORD MajorOperatingSystemVersion; *// 要求的最低OS主版本*

WORD MinorOperatingSystemVersion; *// 要求的最低OS次版本*

WORD MajorImageVersion; *// 映像主版本号（用户定义）*

WORD MinorImageVersion; *// 映像次版本号（用户定义）*

WORD MajorSubsystemVersion; *// 子系统主版本（通常4=Win95）*

WORD MinorSubsystemVersion; *// 子系统次版本*

DWORD Win32VersionValue; *// 保留（必须为0）*

DWORD SizeOfImage; *// 映像在内存中的总大小*

DWORD SizeOfHeaders; *// 所有头部的总大小（对齐后）*

DWORD CheckSum; *// 校验和（驱动/DLL常用）*

WORD Subsystem; *// 子系统类型（1=Native，2=GUI，3=CUI）*

WORD DllCharacteristics; *// DLL属性（如ASLR/DEP）*

DWORD SizeOfStackReserve; *// 初始保留的栈大小*

DWORD SizeOfStackCommit; *// 初始提交的栈大小*

DWORD SizeOfHeapReserve; *// 初始保留的堆大小*

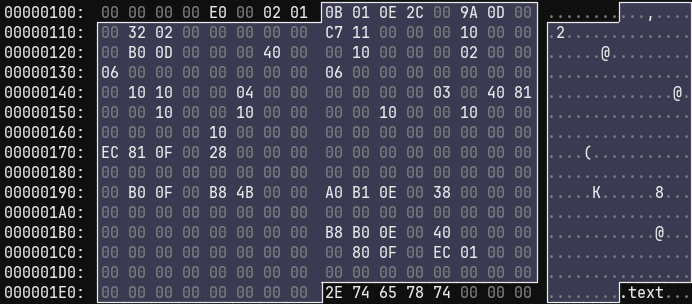
DWORD SizeOfHeapCommit; *// 初始提交的堆大小*

DWORD LoaderFlags; *// 保留（已废弃）*

DWORD NumberOfRvaAndSizes; *// 数据目录项数（通常16）*

IMAGE\_DATA\_DIRECTORY DataDirectory[IMAGE\_NUMBEROF\_DIRECTORY\_ENTRIES]; *// 数据目录表*

} IMAGE\_OPTIONAL\_HEADER32, \*PIMAGE\_OPTIONAL\_HEADER32;



typedef struct \_IMAGE\_OPTIONAL\_HEADER64 {

*// 标准字段（与32位类似）*

WORD Magic; *// 标识：0x20B=64位*

BYTE MajorLinkerVersion;

BYTE MinorLinkerVersion;

DWORD SizeOfCode;

DWORD SizeOfInitializedData;

DWORD SizeOfUninitializedData;

DWORD AddressOfEntryPoint; *// 入口点RVA*

DWORD BaseOfCode; *// 代码段起始RVA*

*// BaseOfData 字段在64位中不存在！*

*// NT扩展字段*

ULONGLONG ImageBase; *// 64位优先加载地址*

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;

ULONGLONG SizeOfStackReserve; *// 64位栈/堆大小*

ULONGLONG SizeOfStackCommit;

ULONGLONG SizeOfHeapReserve;

ULONGLONG SizeOfHeapCommit;

DWORD LoaderFlags;

DWORD NumberOfRvaAndSizes; *// 数据目录项数（通常16）*

IMAGE\_DATA\_DIRECTORY DataDirectory[IMAGE\_NUMBEROF\_DIRECTORY\_ENTRIES]; *// 数据目录表*

} IMAGE\_OPTIONAL\_HEADER64, \*PIMAGE\_OPTIONAL\_HEADER64;

节区头

每个节区头固定40字节。

typedef struct \_IMAGE\_SECTION\_HEADER {

BYTE Name[8]; *// 节区名（如".text"）*

union {

DWORD PhysicalAddress;

DWORD VirtualSize; *// 内存中的实际大小*

} Misc;

DWORD VirtualAddress; *// 内存中的RVA*

DWORD SizeOfRawData; *// 文件中的大小*

DWORD PointerToRawData; *// 文件中的偏移*

DWORD PointerToRelocations; *// 重定位表偏移*

DWORD PointerToLinenumbers; *// 行号表偏移*

WORD NumberOfRelocations; *// 重定位项数*

WORD NumberOfLinenumbers; *// 行号项数*

DWORD Characteristics; *// 节区属性（可读/可写/可执行等）*

} IMAGE\_SECTION\_HEADER, \*PIMAGE\_SECTION\_HEADER;





