



南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异



汇编语言与逆向技术

第7章 PE文件结构

王志

zwang@nankai.edu.cn

Updated on: 2022-10-27

南开大学 网络空间安全学院

2022-2023学年



允公允能 日新月异

```
.data
    lpHexString    db "0123456789ABCDEF"
.code
    Dec2Hex proc dwValue:DWORD, lpBuffer:DWORD
        mov edi, lpBuffer
        mov eax, dwValue
        mov ecx, 8
    repeat:
        mov esi, eax
        and esi, 0F0000000h
        shr esi, 28
        movzx edx, byte ptr [lpHexString+esi]
        mov byte ptr [edi], dl
        shl eax, 4
        inc edi
    loop repeat
    ret
    Dec2Hex endp
```



南开大学
Nankai University



允公允能 日新月异

本章知识点

1. 可执行文件
2. PE文件基本概念
3. DOS文件头
4. PE文件头





南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异



1. 可执行文件



允公允能 日新月异

可执行文件

- 可执行文件 (executable file)
 - 可以由**操作系统**进行**加载**、**执行**的文件
 - 在不同的操作系统环境下，可执行文件的格式不一样
 - 二进制文件，不同于txt、word、excel等文本文件





允公允能 日新月异

可执行文件格式

- Windows系统可执行文件使用**PE**文件格式（Portable Executable）
- Linux系统可执行文件使用**ELF**文件格式（Executable and Linkable Format）



南开大学
Nankai University



允公允能 日新月异

Windows系统可执行文件

- 在Windows操作系统下，可执行程序可以是.com文件、.exe文件、.sys文件、.dll文件、.scr文件等类型文件



南开大学
Nankai University



允公允能 日新月异

.com文件

- .com文件在IBM PC早期出现，格式主要用于命令行应用程序、最大65,280字节
- 与MS-DOS操作系统的可执行文件兼容

 mode.com C:\Windows\System32	修改日期: 2018/9/15 15:29 大小: 30.5 KB
 tree.com C:\Windows\System32	修改日期: 2018/9/15 15:29 大小: 19.5 KB
 chcp.com C:\Windows\System32	修改日期: 2018/9/15 15:29 大小: 14.0 KB
 more.com C:\Windows\System32	修改日期: 2018/9/15 15:29 大小: 28.0 KB





允公允能 日新月异

.exe、.dll、.sys可执行文件

- .exe,.dll,.sys文件使用的是PE文件结构
- PE（**Portable Executable** File Format）可移植可执行文件结构
- 理解PE文件结构是逆向技术的**基础**

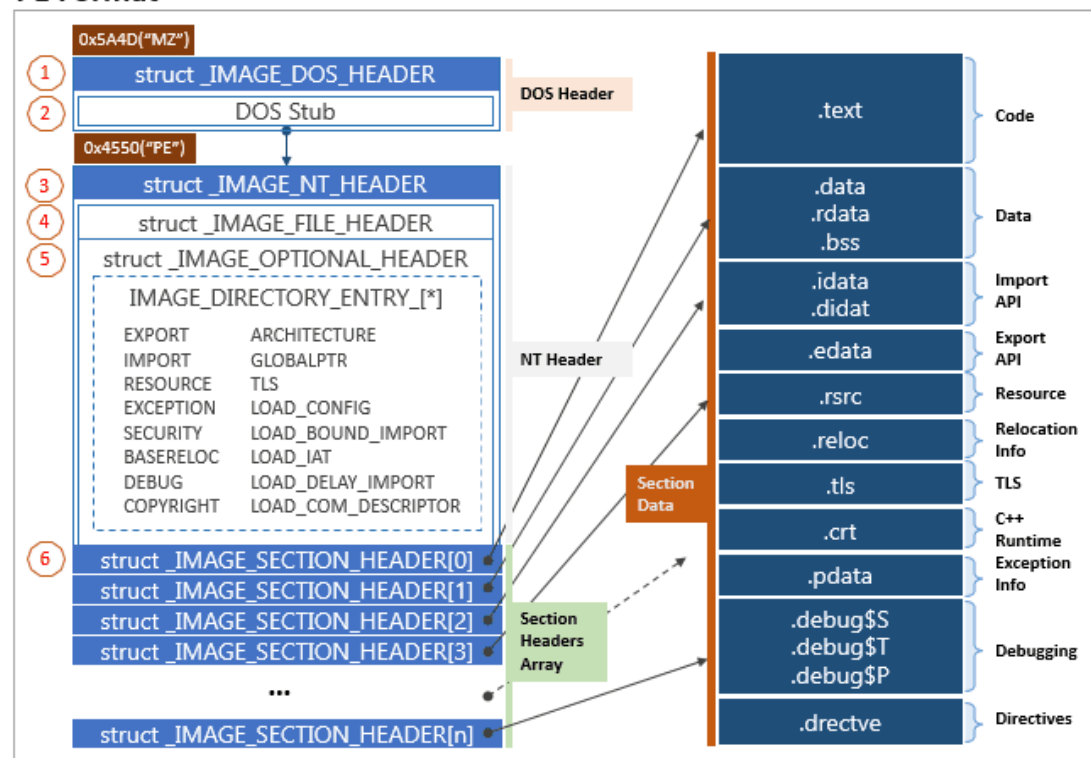


南开大学
Nankai University

PE结构的二进制文件

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68!...L!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....\$......
00000080	C5	7E	C6	DE	81	1F	A8	8D	81	1F	A8	8D	81	1F	A8	8D	.~.....
00000090	0F	00	BB	8D	87	1F	A8	8D	7D	3F	BA	8D	82	1F	A8	8D}?......
000000A0	52	69	63	68	81	1F	A8	8D	00	00	00	00	00	00	00	00	Rich.....
000000B0	50	45	00	00	4C	01	03	00	15	9C	82	5E	00	00	00	00	PE..L.....^....
000000C0	00	00	00	00	E0	00	0F	01	0B	01	05	0C	00	02	00	00
000000D0	00	04	00	00	00	00	00	00	00	10	00	00	00	10	00	00@.....
000000E0	00	20	00	00	00	00	40	00	00	10	00	00	00	02	00	00
000000F0	04	00	00	00	00	00	00	00	04	00	00	00	00	00	00	00@.....
00000100	00	40	00	00	00	04	00	00	00	00	00	00	03	00	00	00
00000110	00	00	10	00	00	10	00	00	00	00	10	00	00	10	00	00
00000120	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00
00000130	10	20	00	00	28	00	00	00	00	00	00	00	00	00	00	00(.....
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

PE Format





南開大學

NANKAI UNIVERSITY, P.R. CHINA 1919

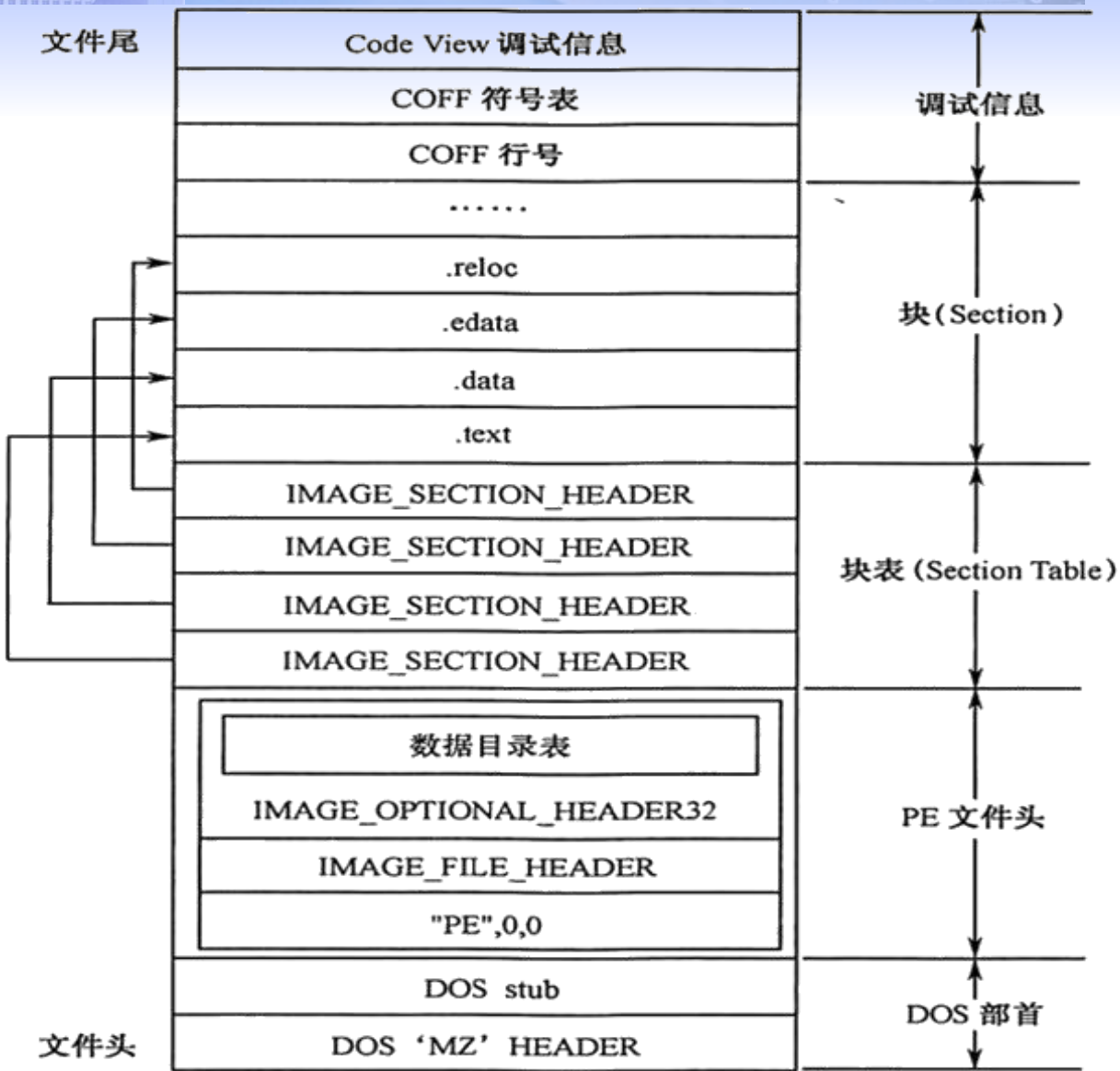
允公允能 日新月異



2. PE文件基本概念



异新月日能允公允





允公允能 日新月异

PE文件结构

- PE文件使用的是一个平面地址空间
 - 所有代码和数据都合并在一起，组成了一个很大的结构
- 文件的内容被分割为不同的节 (Section，也叫做块、区块等)



南开大学
Nankai University



允公允能 日新月异

节

- 代码节、数据节
- 各个节按页边界对齐
- 节是一个连续结构，没有大小限制
- 每个节都有自己的内存属性



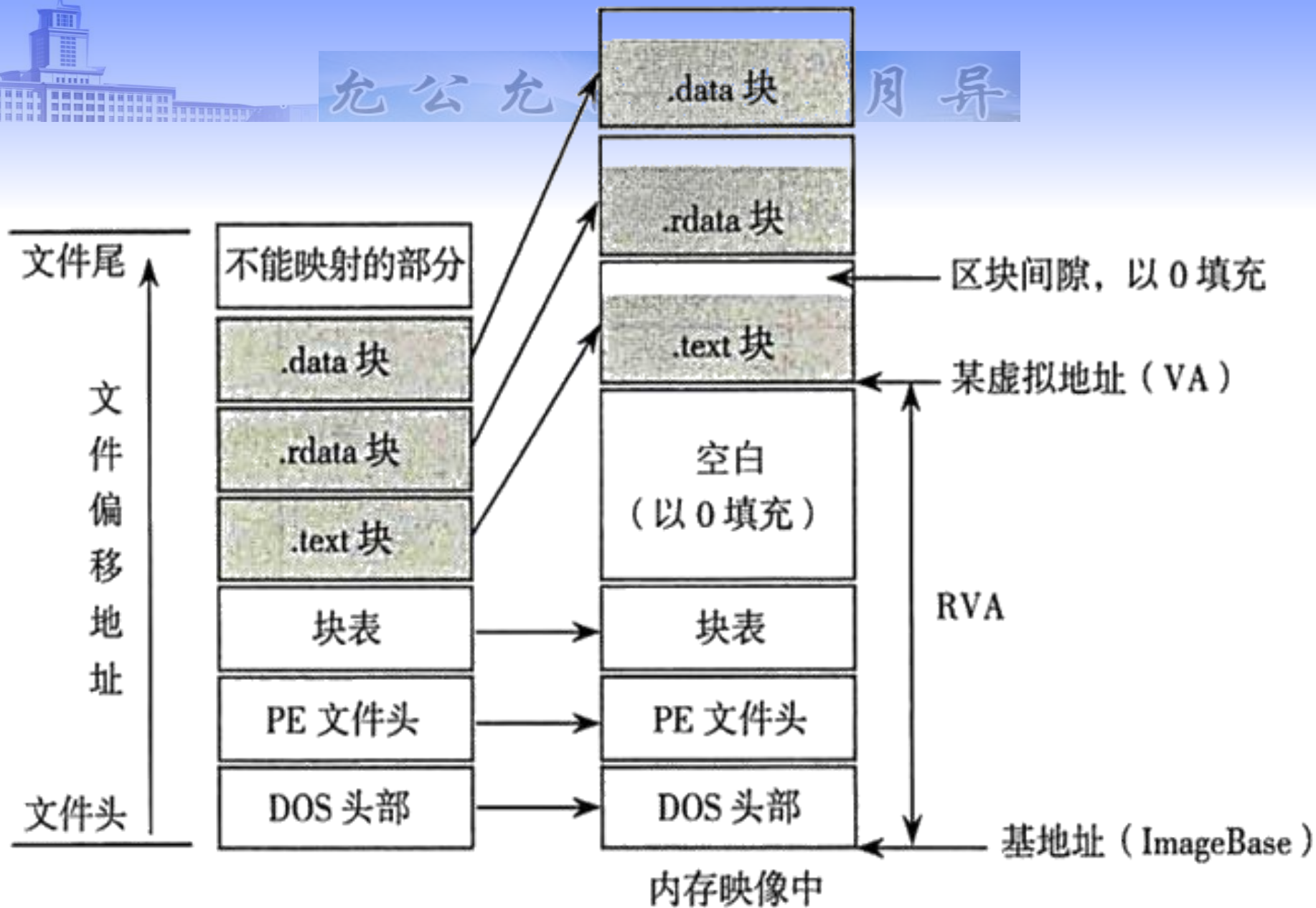


图 11.2 PE 文件磁盘与内存映像结构图



允公允能 日新月异

虚拟内存地址空间

- 每个程序都有自己的**虚拟内存地址空间**，虚拟空间的内存地址称为**虚拟地址** (Virtual Address, VA)
 - 4GB
 - 不同进程的虚拟地址空间是相互**隔离**的





模块

00400000	00001000	proc		PE 文件头
00401000	00001000	proc	.text	码
00402000	00001000	proc	.rdata	入表
00403000	00001000	proc	.data	据
00410000	000C5000			
004E0000	00006000			
006E0000	00005000			
75580000	00001000	KERNEL32		PE 文件头
75590000	00064000	KERNEL32	.text	码
75600000	0002F000	KERNEL32	.rdata	入表, 输出表
75630000	00001000	KERNEL32	.data	据
75640000	00001000	KERNEL32	.rsrc	源
75650000	00005000	KERNEL32	.reloc	定位
76830000	00001000	KERNELBASE		PE 文件头
76831000	001C3000	KERNELBASE	.text	码, 输出表
769F4000	00004000	KERNELBASE	.data	据
769F8000	00006000	KERNELBASE	.idata	入表
769FE000	00001000	KERNELBASE	.didat	
769FF000	00001000	KERNELBASE	.rsrc	源
76A00000	0002A000	KERNELBASE	.reloc	定位
77400000	00001000	USER32		PE 文件头





允公允能 日新月异

模块

- 当PE文件通过Windows加载器载入内存后，内存中的版本称为**模块** (Module)
 - 映射文件的起始地址称为**模块句柄** (hModule)
 - 初始内存地址也称为**基地址** (ImageBase)





允公允能 日新月异

模块

- 内存中的**模块**代表进程将这个可执行文件所需要的代码、数据、资源、输入表、输出表及其它有用的数据结构都放在一个连续的内存节中





允公允能 日新月异

模块句柄

- Windows将Module的基地址作为Module的实例句柄 (Instance Handle, 即Hinstance)。
- GetModuleHandle获得DLL句柄, 通过句柄访问DLL Module的内容

```
HMODULE GetModuleHandle(LPCTSTR lpModuleName);
```





虚拟地址

00401000	BE 00000000	MOV ESI, 0		寄存器 (FPU)
00401005	B9 0E000000	MOV ECX, 0E		EAX 0019FFCC
0040100A	8A86 00304000	MOV AL, BYTE PTR DS:[ESI+403000]	ASCII "Hello World", CR, LF	ECX 00401000 pr
00401010	8886 0E304000	MOV BYTE PTR DS:[ESI+40300E], AL		EDX 00401000 pr
00401016	46	INC ESI		EBX 002F1000
00401017	E2 F1	LOOP SHORT 0040100A		ESP 0019FF74
00401019	68 00304000	PUSH OFFSET 00403000	ASCII "Hello World", CR, LF	EBP 0019FF80
0040101E	E8 11000000	CALL 00401034		ESI 00401000 pr
00401023	68 0E304000	PUSH OFFSET 0040300E		EDI 00401000 pr
00401028	E8 07000000	CALL 00401034		EIP 00401000 pr
0040102D	6A 00	PUSH 0		C 0 ES 002B 32
0040102F	E8 AE000000	CALL <JMP.&kernel32.ExitProcess>	跳转至 KERNEL32.ExitProcess	P 1 CS 0023 32
00401034	55	PUSH EBP		A 0 SS 002B 32
00401035	8BEC	MOV EBP, ESP		Z 1 DS 002B 32
00401037	83C4 F4	ADD ESP, -0C		S 0 FS 0053 32
0040103A	6A F5	PUSH -0B		T 0 GS 002B 32
0040103C	E8 A7000000	CALL <JMP.&kernel32.GetStdHandle>	跳转至 KERNEL32.GetStdHandle	D 0
00401041	8945 EC	MOV DWORD PTR SS:[EBP-4], EAX		O 0 LastErr 00
Imm=0				
ESI=00401000 (proc.<ModuleEntryPoint>)				
地址	十六进制数据	多字节 (ANSI/OEM - 简体中文)	0019FF74	755A0419
00403000	48 65 6C 6C 6F 20 57 6F 72 6C 64 0D 0A 00 00 00	Hel lo Wor ld	0019FF78	002F1000
00403010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0019FF7C	755A0400
00403020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0019FF80	0019FFDC
00403030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0019FF84	7750662D
00403040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0019FF88	002F1000
00403050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0019FF8C	51667E66
00403060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0019FF90	00000000
00403070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0019FF94	00000000
00403080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0019FF98	002F1000
00403090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0019FF9C	00000000
004030A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0019FFA0	00000000
004030B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0019FFA4	00000000





允公允能 日新月异

相对虚拟地址

- 在可执行文件中，有许多地方需要内存地址
 - 例如，引用全局变量时需要指定它的地址
- PE文件有一个首选的载入地址(基地址)
- PE文件可以载入到进程空间任何地方





允公允能 日新月异

相对虚拟地址

- 为了避免绝对内存地址，引入了相对虚拟地址 (Relative Virtual Address, RVA) 的概念
 - RVA是相对于PE文件载入地址的偏移位置



南开大学
Nankai University



允公允能 日新月异

相对虚拟地址

- 假设一个EXE文件从400000h处载入内存
- 代码节起始地址为401000h，代码节起始地址的RVA计算方法如下：
 - 目标地址401000h - 载入地址400000h = RVA 1000h





允公允能 日新月异

相对虚拟地址

- 将RVA转换成虚拟地址VA的过程
 - 用实际的载入地址ImageBase加相对虚拟地址RVA
 - 虚拟地址 (VA) = 基地址 (ImageBase) + 相对虚拟地址 (RVA)





允公允能 日新月异

文件偏移地址

- PE文件储存在磁盘中，某个数据的位置相对于文件头的偏移量称为文件偏移地址 (File Offset) 或物理地址 (RAW Offset)。
 - 文件偏移地址从PE文件的第1个字节开始计数，起始值为0。



南开大学
Nankai University



PE文件

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	B0	00	00	00
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68!..L.!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....\$......
00000080	C5	7E	C6	DE	81	1F	A8	8D	81	1F	A8	8D	81	1F	A8	8D	.~.....
00000090	0F	00	BB	8D	87	1F	A8	8D	7D	3F	BA	8D	82	1F	A8	8D}?......
000000A0	52	69	63	68	81	1F	A8	8D	00	00	00	00	00	00	00	00	Rich.....
000000B0	50	45	00	00	4C	01	03	00	15	9C	82	5E	00	00	00	00	PE..L.....^....
000000C0	00	00	00	00	E0	00	0F	01	0B	01	05	0C	00	02	00	00
000000D0	00	04	00	00	00	00	00	00	00	10	00	00	00	10	00	00
000000E0	00	20	00	00	00	00	40	00	00	10	00	00	00	02	00	00@.....
000000F0	04	00	00	00	00	00	00	00	04	00	00	00	00	00	00	00
00000100	00	40	00	00	00	04	00	00	00	00	00	00	03	00	00	00	.@.....
00000110	00	00	10	00	00	10	00	00	00	00	10	00	00	10	00	00
00000120	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00
00000130	10	20	00	00	28	00	00	00	00	00	00	00	00	00	00	00	. ..(.
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00





南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异



DOS文件头



允公允能 日新月异

DOS文件头

- 每个PE文件都是以一个16位的DOS程序开始的
- DOS MZ头与DOS stub合称为DOS文件头



IMAGE_DOS_HEADER

```

IMAGE_DOS_HEADER_STRUCT{
+0h  e_magic      WORD ?      ;DOS 可执行文件标记 "MZ"
+2h  e_cblp       WORD ?
+4h  e_cp         WORD ?
+6h  e_crlc       WORD ?
+8h  e_cparhdr    WORD ?
+0ah  e_minalloc  WORD ?
+0ch  e_maxalloc  WORD ?
+0eh  e_ss        WORD ?
+10h  e_sp        WORD ?
+12h  e_csum      WORD ?
+14h  e_ip        WORD ?      ;DOS 代码入口 IP
+16h  e_cs        WORD ?      ;DOS 代码入口 CS
+18h  e_lfarlc    WORD ?
+1ah  e_ovno      WORD ?
+1ch  e_res       WORD 4 dup(?)
+24h  e_oemid     WORD ?
+26h  e_oeminfo   WORD ?
+28h  e_res2      WORD 10 dup(?)
+3ch  e_lfanew    DWORD ?      ;指向 PE 文件头"PE",0,0
} IMAGE_DOS_HEADER_ENDS
    
```



允公允能 日新月异

DOS文件头

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	B0	00	00	00
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68!...L!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....\$.....
00000080	C5	7E	C6	DE	81	1F	A8	8D	81	1F	A8	8D	81	1F	A8	8D	.~.....
00000090	0F	00	BB	8D	87	1F	A8	8D	7D	3F	BA	8D	82	1F	A8	8D}?......
000000A0	52	69	63	68	81	1F	A8	8D	00	00	00	00	00	00	00	00	Rich.....
000000B0	50	45	00	00	4C	01	03	00	15	9C	82	5E	00	00	00	00	PE..L.....^.....





南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异



PE文件头



允公允能 日新月异

PE文件头

- PE文件头 (PE Header) 紧跟在DOS stub的后面
- IMAGE_DOS_HEADER结构的e_lfanew字段定位PE Header的起始偏移量，加上基址，得到PE文件头的指针
 - $PNTHeader = ImageBase + DosHeader \rightarrow e_lfanew$



南开大学
Nankai University



PE文件头

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	B0	00	00	00
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68!..L.!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....\$......
00000080	C5	7E	C6	DE	81	1F	A8	8D	81	1F	A8	8D	81	1F	A8	8D	.~.....
00000090	0F	00	BB	8D	87	1F	A8	8D	7D	3F	BA	8D	82	1F	A8	8D}?......
000000A0	52	69	63	68	81	1F	A8	8D	00	00	00	00	00	00	00	00	Rich.....
000000B0	50	45	00	00	4C	01	03	00	15	9C	82	5E	00	00	00	00	PE..L.....^.....
000000C0	00	00	00	00	E0	00	0F	01	0B	01	05	0C	00	02	00	00
000000D0	00	04	00	00	00	00	00	00	00	10	00	00	00	10	00	00
.....





PE文件头

IMAGE_NT_HEADERS32 structure (winnt.h)

04/02/2021 • 2 minutes to read

Represents the PE header format.

- Signature
- FileHeader
- OptionalHeader

Syntax

C++

Copy

```
typedef struct _IMAGE_NT_HEADERS {  
    DWORD Signature;  
    IMAGE_FILE_HEADER FileHeader;  
    IMAGE_OPTIONAL_HEADER32 OptionalHeader;  
} IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;
```

https://docs.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-image_nt_headers32





允公允能 日新月异

Signature

- 在一个有效的PE文件，Signature字段的值是00004550h
 - ASCII字符是“PE\0\0”

```
#define IMAGE_NT_SIGNATURE
```

```
0x00004550
```



南開大學
Nankai University

FileHeader

- IMAGE_FILE_HEADER结构中记录了PE文件的一些基本信息，并指出了IMAGE_OPTIONAL_HEADER的大小

Syntax

IMAGE_FILE_HEADER STRUCT

+04h Machine	WORD	?	;运行平台
+06h NumberOfSections	WORD	?	;文件的区块数
+08h TimeDateStamp	DWORD	?	;文件创建日期和时间
+0Ch PointerToSymbolTable	DWORD	?	;指向符号表（用于调试）
+10h NumberOfSymbols	DWORD	?	;符号表中符号的个数（用于调试）
+14h SizeOfOptionalHeader	WORD	?	;IMAGE_OPTIONAL_HEADER32 结构的
+16h Characteristics	WORD	?	;文件属性

IMAGE_FILE_HEADER ENDS

C++

```
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;
```



FileHeader

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	B0	00	00	00
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68!...L.!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....\$.....
00000080	C5	7E	C6	DE	81	1F	A8	8D	81	1F	A8	8D	81	1F	A8	8D	.~.....
00000090	0F	00	BB	8D	87	1F	A8	8D	7D	3F	BA	8D	82	1F	A8	8D}?......
000000A0	52	69	63	68	81	1F	A8	8D	00	00	00	00	00	00	00	00	Rich.....
000000B0	50	45	00	00	4C	01	03	00	15	9C	82	5E	00	00	00	00	PE..L.....^.....
000000C0	00	00	00	00	E0	00	0F	01	0B	01	05	0C	00	02	00	00
000000D0	00	04	00	00	00	00	00	00	00	10	00	00	00	10	00	00
000000E0	00	20	00	00	00	00	40	00	00	10	00	00	00	02	00	00@.....
000000F0	04	00	00	00	00	00	00	00	04	00	00	00	00	00	00	00





Machine

- Machine

- 可执行文件的目标CPU类型
- PE文件可以在多种CPU机器上使用，不同平台上指令的机器码不同。

Value	Meaning
IMAGE_FILE_MACHINE_I386 0x014c	x86
IMAGE_FILE_MACHINE_IA64 0x0200	Intel Itanium
IMAGE_FILE_MACHINE_AMD64 0x8664	x64

https://docs.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-image_file_header





允公允能 日新月异

NumberOfSections

- NumberOfSections记录节 (Section) 的数量
 - 节表紧跟在IMAGE_NT_HEADERS后面定义





允公允能 日新月异

TimeStamp

- TimeDateStamp
 - 文件的创建时间
 - 1970年1月1日到创建该文件的所有的秒数

现在: 1586161977 控制: ☒ 停止

时间戳 秒(s) ▼ 北京时间

时间 北京时间 秒(s) ▼

获取当前时间戳



南开大学
Nankai University



Characteristics

特征值	含义
0001h	文件中不存在重定位信息
0002h	文件可执行。如果为 0，一般是链接时出问题了
0004h	行号信息被移去
0008h	符号信息被移去
0020h	应用程序可以处理超过 2GB 的地址。该功能是从 NT SP3 开始被支持的。因为大部分数据库服务器需要很大的内存，而 NT 仅提供 2GB 给应用程序，所以从 NT SP3 开始，通过加载 /3GB 参数，可以使应用程序被分配 2~3GB 区域的地址，而该处原来属于系统内存区
0080h	处理机的低位字节是相反的
0100h	目标平台为 32 位机器
0200h	.DBG 文件的调试信息被移去
0400h	如果映像文件在可移动介质中，则先复制到交换文件中再运行
0800h	如果映像文件在网络中，则复制到交换文件后才运行
1000h	系统文件
2000h	文件是 DLL 文件
4000h	文件只能运行在单处理器上
8000h	处理机的高位字节是相反的





允公允能 日新月异

IMAGE_OPTIONAL_HEADER

- 可选映像头 (IMAGE_OPTIONAL_HEADER) 结构中定义了更多的PE文件数据
- IMAGE_FILE_HEADER和IMAGE_OPTIONAL_HEADER合在一起构成PE文件头



南开大学
Nankai University



IMAGE_OPTIONAL_HEADER

```
//IMAGE_OPTIONAL_HEADER结构(可选映像头)
typedef struct _IMAGE_OPTIONAL_HEADER {
    //
    // Standard fields.
    //
    WORD    Magic;    //幻数, 一般为10BH
    BYTE    MajorLinkerVersion;    //链接程序的主版本号
    BYTE    MinorLinkerVersion;    //链接程序的次版本号
    DWORD   SizeOfCode;    //代码段大小
    DWORD   SizeOfInitializedData;    //已初始化数据块的大小
    DWORD   SizeOfUninitializedData;    //未初始化数据库的大小
    DWORD   AddressOfEntryPoint;    //程序开始执行的入口地址,这是一个RVA (相对虚拟地址)
    DWORD   BaseOfCode;    //代码段的起始RVA 一般来说 是 1000h
    DWORD   BaseOfData;    //数据段的起始RVA
    //
}
```

```

//
// NT additional fields.
//
DWORD   ImageBase; //可执行文件默认装入的基地址
DWORD   SectionAlignment; //内存中块的对齐值 (默认的块对齐值为1000H, 4KB个字节)
DWORD   FileAlignment; //文件中块的对齐值 (默认值为200H字节, 为了保证块总是从磁盘的扇区开始的)
WORD     MajorOperatingSystemVersion; //要求操作系统的最低版本号的主版本号
WORD     MinorOperatingSystemVersion; //要求操作系统的最低版本号的次版本号
WORD     MajorImageVersion; //该可执行文件的主版本号
WORD     MinorImageVersion; //该可执行文件的次版本号
WORD     MajorSubsystemVersion; //要求最低之子系统版本的主版本号 默认 0004
WORD     MinorSubsystemVersion; //要求最低之子系统版本的次版本号 默认 0000
DWORD    Win32VersionValue; //保留字 默认00000000
DWORD    SizeOfImage; //映像装入内存后的总尺寸 一般为00004000 映射到内存中一个块1000内存
DWORD    SizeOfHeaders; //部首及块表的大小
DWORD    CheckSum; //CRC检验和 一般为00000000
WORD     Subsystem; //程序使用的用户接口子系统
WORD     DllCharacteristics; //DLLmain函数何时被调用, 默认为0
DWORD    SizeOfStackReserve; //初始化时堆栈大小
DWORD    SizeOfStackCommit; //初始化时实际提交的堆栈大小
DWORD    SizeOfHeapReserve; //初始化时保留的堆大小
DWORD    SizeOfHeapCommit; //初始化时实际提交的对大小
DWORD    LoaderFlags; //与调试有关, 默认为0
DWORD    NumberOfRvaAndSizes; //数据目录结构的数目
IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES]; //数据目录表
}

```





IMAGE_OPTIONAL_HEADER

- Magic (**WORD**) : 这是一个标记字, 说明文件是ROM映像 (0107h) 还是普通可执行的映像 (010Bh)
 - 010Bh
- MajorLinkerVersion (**BYTE**) : 链接程序的主版本号
 - 05h
- MinorLinkerVersion (**BYTE**) : 链接程序的次版本号
 - 0Ch





允公允能 日新月异

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	M	Z
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000030	00	00	00	00	00	00	00	00	00	00	00	00	B0	00	00	00	
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	i	s		p	r	o	g	r	a	m		c	a	n	n	
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t		b	e		r	u	n		i	n		D	O	S	
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	m	o	d	e	
00000080	C5	7E	C6	DE	81	1F	A8	8D	81	1F	A8	8D	81	1F	A8	8D	.	~	
00000090	0F	00	BB	8D	87	1F	A8	8D	7D	3F	BA	8D	82	1F	A8	8D	
000000A0	52	69	63	68	81	1F	A8	8D	00	00	00	00	00	00	00	00	R	i	c	h	
000000B0	50	45	00	00	4C	01	03	00	15	9C	82	5E	00	00	00	00	P	E	
000000C0	00	00	00	00	E0	00	0F	01	0B	01	05	0C	00	02	00	00	
000000D0	00	04	00	00	00	00	00	00	00	10	00	00	00	10	00	00	
000000E0	00	20	00	00	00	00	40	00	00	10	00	00	00	02	00	00	
000000F0	04	00	00	00	00	00	00	00	04	00	00	00	00	00	00	00	





允公允能 日新月异

IMAGE_OPTIONAL_HEADER

- SizeOfCode(**DWORD**): 代码段的大小
 - 00000200h
- SizeOfInitializedData(**DWORD**): 已初始化数据块的大小:
00000400h
- SizeOfUninitializedData(**DWORD**): 未初始化数据块的大小:
00000000h





允公允能 日新月异

IMAGE_OPTIONAL_HEADER

- AddressOfEntryPoint (DWORD) 程序执行的入口RVA: 00001000h
- BaseOfCode (DWORD) 代码段的起始RVA: 00001000h
- BaseOfData (DWORD) 数据段的起始RVA: 00002000h
- ImageBase (DWORD) 文件在内存中的载入地址
 - 00400000h





允公允能 日新月异

IMAGE_OPTIONAL_HEADER

- SectionAlignment(**DWORD**): 内存上区块间的对齐大小
 - 00001000h
 - 多少字节?
- FileAlignment(DWORD): 硬盘上区块间的对齐大小
 - 00000200h
 - 多少字节?





允公允能 日新月异

IMAGE_OPTIONAL_HEADER

- MajorOperatingSystemVersion (WORD)
- MinorOperatingSystemVersion (WORD)
 - PE文件执行所需要的Windows系统最低版本号
 - 0004、0000
 - Windows NT 的内部版本号是 4.0
 - Win2000是5.0、XP是5.1、Vista是6.0、Win7是6.1、Win8是6.2、Win10是10.0、Win11是10.0





允公允能 日新月异

IMAGE_OPTIONAL_HEADER

- MajorImageVersion (WORD)
- MinorImageVersion (WORD)
 - 程序的主版本号 and 次版本号
 - 由程序员自己定义
 - 0000 0000





允公允能 日新月异

IMAGE_OPTIONAL_HEADER

- MajorSubsystemVersion (WORD)
- MinorSubsystemVersion (WORD)
 - PE文件所要求的子系统的版本号
 - 0004 0000
- Win32VersionValue (DWORD)：通常被设置为0





允公允能 日新月异

IMAGE_OPTIONAL_HEADER

- SizeOfImage (**DWORD**) : 映像载入内存后的总大小, 是指载入文件从ImageBase到最后一个块的大小
 - 00004000h
- SizeOfHeaders (**DWORD**) : MS-DOS头部、PE文件头、区块表的总尺寸
 - 00000400h





允公允能 日新月异

IMAGE_OPTIONAL_HEADER

- SizeOfStackReserve (DWORD)
- SizeOfStackCommit (DWORD)
- SizeOfHeapReserve (DWORD)
- SizeOfHeapCommit (DWORD)
 - 栈、堆的设置信息
- LoaderFlags (DWORD) : 与调试有关



南开大学
Nankai University

IMAGE_DATA_DIRECTORY 数据目录

- NumberOfRvaAndSizes (**DWORD**) : 数据目录的项数
- DataDirectory[16]:数据目录表

```
#define IMAGE_DIRECTORY_ENTRY_EXPORT      0 // 导出表
#define IMAGE_DIRECTORY_ENTRY_IMPORT      1 // 导入表
#define IMAGE_DIRECTORY_ENTRY_RESOURCE    2 // 资源
#define IMAGE_DIRECTORY_ENTRY_EXCEPTION   3 // 异常
#define IMAGE_DIRECTORY_ENTRY_SECURITY    4 // 安全
#define IMAGE_DIRECTORY_ENTRY_BASERELOC    5 // 重定位表
#define IMAGE_DIRECTORY_ENTRY_DEBUG        6 // 调试信息
#define IMAGE_DIRECTORY_ENTRY_COPYRIGHT    7 // (X86 usage)
#define IMAGE_DIRECTORY_ENTRY_ARCHITECTURE 7 // 版权信息
#define IMAGE_DIRECTORY_ENTRY_GLOBALPTR    8 // RVA of GP
#define IMAGE_DIRECTORY_ENTRY_TLS          9 // TLS Directory
#define IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG 10 // Load Configuration Directory
#define IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT 11 // Bound Import Directory in headers
#define IMAGE_DIRECTORY_ENTRY_IAT          12 // 导入函数地址表
#define IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT 13 // Delay Load Import Descriptors
#define IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR 14 // COM Runtime descriptor
```

IMAGE_DATA_DIRECTORY 数据目录

Syntax

C++

```
typedef struct _IMAGE_DATA_DIRECTORY {  
    DWORD VirtualAddress;  
    DWORD Size;  
} IMAGE_DATA_DIRECTORY, *PIMAGE_DATA_DIRECTORY;
```

- VirtualAddress
 - 该目录的相对虚拟地址
 - RVA
- Size
 - 该目录的大小



允公允能 日新月异

数据目录

- 定位PE文件的导入表、导出表、资源的时候，需要用到数据目录



南开大学
Nankai University



允公允能 日新月异

本章知识点

1. 可执行文件
2. PE文件基本概念
3. DOS文件头
4. PE文件头



南开大学
Nankai University



南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异



汇编语言与逆向技术

第7章 PE文件结构

王志

zwang@nankai.edu.cn

Updated on: 2022-10-27

南开大学 网络空间安全学院

2022-2023学年