PE/PE32+ File Structure

0x00

- DOS HEADER
- DOS STUB → THIS PROGRAM CANNOT BE RUN IN DOS MODE
- PE HEADER
- OPTIONAL HEADER
- SECTION HEADERS
- SECTIONS

0x00	WORD	MAGIC	FILE TYPE INDICATOR
0x02	WORD	CBLP	NUMBER OF BYTES ON LAST PAGE OF FILE
0x04	WORD	CP	NUMBER OF PAGES
0x06	WORD	CRLC	NUMBER OF RELOCATIONS IN TABLE
0x08	WORD	CPARHDR	HEADER SIZE
0x0A	WORD	MINALLOC	MINIMAL ALLOCATION SIZE
0x0C	WORD	MAXALLOC	REQUESTED ALLOCATION SIZE
0x0E	WORD	SS	INITIAL SS REGISTER STATE
0x10	WORD	SP	INISTIAL SP REGISTER STATE
0x12	WORD	CSUM	CHECKSUM
0x14	WORD	IP	INITIAL IP REGISTER STATE
0x16	WORD	CS	INISTIAL CS REGISTER STATE
0x18	WORD	LFARLC	ABSOLUTE OFFSET OF RELOCATION TABLE
0x1A	WORD	OVNO	OVERLAY BYTES NUMBER
0x24	WORD	OEMID	OEM IDENTIFIER
0x26	WORD	OEMINFO	OEM INFORMATION
0x3C	WORD	LFANEW	OFFSET OF PE HEADER

THIS IS RVA. ADD THIS OFFSET TO BEGINNING OG THE FILE TO GET PE HEADER VIRTUAL ADDRESS

ADDRESS OF PE HEADER + 0x18

0x00	WORD	MAGIC	
0x02	BYTE	MAJOR LINKER VERSION	
0x03	BYTE	MINOR LINKER VERSION	
0x04	DWORD	SIZE OF CODE	
0x08	DWORD	SIZE OF INITIALIZED DATA	
0x0C	DWORD	SIZE OF UNINITIALIZED DATA	
0x10	DWORD	ADDRESS OF ENTRY POINT	
0x14	DWORD	BASE OF CODE	HERE'S THE .TEXT SECTION
0x18	ULONG	IMAGEBASE	
0x20	DWORD	SECTION ALIGNMENT	
0x24	DWORD	FILE ALIGNMENT	
0x28	WORD	MAJOR OS VERSION	
0x2A	WORD	MINOR OS VERSION	
0x2C	WORD	MAJOR IMAGE VERSION	
0x2E	WORD	MINOR IMAGE VERSION	
0x30	WORD	MAJOR SUBSYSTEM VERSION	
0x32	WORD	MINOR SUBSYSTEM VERSION	
0x34	DWORD	WIN32 VERSION VALUE	
0x38	DWORD	SIZE OF IMAGE	
0x3C	DWORD	SIZE OF HEADERS	
0x40	DWORD	CHECKSUM	
0x44	WORD	SUBSYSTEM	
0x46	WORD	DLL CHARACTERISTICS	
0x48	ULONG	SIZE OF STACK RESERVE	
0x50	ULONG	SIZE OF STACK COMMIT	
0x58	ULONG	SIZE OF HEAP RESERVE	
0x60	ULONG	SIZE OF HEAP COMMIT	
0x68	DWORD	LOADER FLAGS	
0x6C	DWORD	NUMBER OF RVA AND SIZES	
0x70	STRUCT	IMAGE_DATA_DIRECTORY[16]	

TABLE OF POINTERS AND SIZES YOU CAN ITERATE THROUGH THESE RVA TO ACCESS MODIFY PE PROPERTIES ALREADY LOADED TO VIRTUAL MEMORY

0x00	DWORD	MAGIC	PE\0\0
0x04	WORD	MACHINE	ARCHITECTURE
0x06	WORD	SECTIONS	SECTIONS NUMBER
0x08	DWORD	TIMESTAMP	COMPILATION TIME
0x0C	DWORD	SYMBOLS	POINTER TO SYMBOLS TABLE
0x10	DWORD	SYMBOLSNUM	NUMBER OF SYMBOLS
0x14	WORD	SIZEOFOPTHEADER	SIZE OF OPTIONAL HEADER
0x16	WORD	CHARACTERISTICS	FILE PROPERTIES

0x00	DWORD	RVA	EXPORT POINTER
0x04	DWORD	SIZE	EXPORT SIZE
0x08	DWORD	RVA	IMPORT POINTER
0x0C	DWORD	SIZE	IMPORT SIZE
0x10	DWORD	RVA	RESOURCE POINTER
0x14	DWORD	SIZE	RESOURCE SIZE
0x18	DWORD	RVA	EXCEPTION POINTER
0x1C	DWORD	SIZE	EXCEPTION SIZE
0x20	DWORD	RVA	SECURITY POINTER
0x24	DWORD	SIZE	SECURITY SIZE
0x28	DWORD	RVA	BASERELOC POINTER
0x2C	DWORD	SIZE	BASERELOC SIZE
...ETC	...ETC	...ETC	ETC...

GENERAL RULE OF THUIMB WHEN WORKING WITH PE FILE LOADED INTO MEMORY:
ONCE READING 32-BIT RELATIVE VIRTUAL ADDRESS (RVA), CONVERT RVA TO 64BIT UINT AND ADD IT TO IMAGEBASE (VIRTUAL 64-BIT ADDRESS OF PE IN VIRTUAL MEMORY.
THEN, YOU CAN DIRECTLY ACCESS ANY PROPERTY OF PE FILE IN VIRTUAL MEMORY AND MODIFY PE DURING RUNTIME

0xFF...

PE/PE32+ File Structure

0x00

- DOS HEADER
- DOS STUB
- PE HEADER
- OPTIONAL HEADER
- SECTION HEADERS
- SECTIONS

ALLOCATED ONE BY ONE, AFTER OPTIONAL HEADER

0x00	8 BYTES	NAME	SECTION NAME E.G. .TEXT
0x08	DWORD	PHYSICAL ADDRESS	SECTION OFFSET IN PE FILE
0x0C	DWORD	VIRTUAL SIZE	SIZE OF SECTION AFTER LOAD
0x10	DWORD	VIRTUAL ADDRESS	VIRTUAL OFFSET AFTER LOAD
0x14	DWORD	SIZE OF RAW DATA	SIZE OF SECTION IN FILE
0x18	DWORD	RAW DATA PTR	POINTER TO SECTION IN FILE
0x1C	DWORD	RELOC PTR	POINTER TO RELOCATION ENTRIES FOR SECTION
0x20	DWORD	LINES_NUMBER PTR	POINTER TO LINE NUMBERS IF SPECIFIED
0x24	WORD	RELOC NUMBER	NUMBER OF RELOCATIONS IN SECTION
0x26	WORD	LINES_NUMBER NBR	NUMBER OF LINE_NUMBERS IN SECTION
0x28	DWORD	CHARACTERISITCS	DATA ACCESS RULES E.G. R_W,RW ETC.

SECTION SIZE AFTER LOADING TO VIRTUAL MEMORY

ACTUAL RVA OF THE SECTION

DETERMINES DATA ACCESS TO TO SECTION'S MEMORY

ALLOCATED ONE BY ONE, AFTER SECTION HEADERS

CREATION OF INDIVIDUAL SECTIONS ARE COMPILER DEPENDEND.
GCC CREATES NUMBERED SECTIONS WITHOUT A NAME,
MICROSOFT COMPILER IS MORE COMPLIANT WITH THIS INFO.
ALSO, AFORMENTIONED SECTIONS MAY BE ALLOCATED IN ONE SECTION,
E.G. IAT ADDRESS TABLE MAY BE ALLOCATED IN .RDATA SECTION

.TEXT	EXECUTABLE BYTECODE
.RDATA	READONLY DATA E.G. HARDCODED STRINGS
.IDATA	IAT TABLE, BUT MAY BE CONNECTED WITH .RDATA
.EDATA	EAT TABLE, ALSO MAY BE CONNECTED WITH .RDATA
.DATA	READABLE/WRITABLE DATA
.PDATA	CONTAINS EXCEPTIONS TABLES, ADDED IN PE32+ TO HANDLE EXCEPTIONS
.RSRC	CONTAINS ICONS, MENUS ETC.
.RELOC	CONTAINS RELOCATION DATA - USED BY LOADER TO ALLOCATE HARDCODED DATA WITH NEW IMAGEBASE
.BSS	STORES VARIABLES THAT ARE STATIC OR GLOBAL IN CODE, BUT WASN'T INITIALIZED
.DEBUG	CONTAINS DATA REQUIRED FOR SYMBOLIC DEBUGGING

REBASING AND RELOCATIONS:

WHEN LOADING THE PE FILE TO VIRTUAL NEMORY, OPEATING SYSTEM MAY ALLOCATE IT UNDER ARBITRARY CHOSEN VIRTUAL ADDRESS. ESPECIALLY, WHEN ASR IS ENABLED,
THE ALLOCATION ADDRESS WILL BE RANDOMLY ASSIGNED. IN THIS CASE,THE HARDCODED ELEMENTS MUST BE RELOCATED IN VIRTUAL MEMORY.
RELOCATION DATA IS PRESENT IN .RELOC SECTION, AND CONTAINS DATA OFFSET AND SIZE OF THE DATA TO BE RELOCATED.
DATA OFFSET IS THEN BEING ADDED TO DELTA BETWEEN ARBITRARY CHOSEN BASE OF IMAGE AND IMAGE BASE MENTIONED IN OPTIONAL HEADER.

DATA OFFSET IS THEN BEING ADDED TO DELTA BETWEEN ARBITRARY CHOSEN BASE OF IMAGE AND IMAGE BASE MENTIONED IN OPTIONAL HEADER.

ALLOCATING SECTIONS IN RUNTIME:

COPYING SECTIONS FROM THE RAW PE FILE INTO VIRTUAL MEMORY, OPERATING SYSTEM ALLOCATES THE MEMORY ACCORDING TO THE VIRTUAL MEMORY FIELD,
TAKEN FROM SECTION HEADER, MENTIONED ABOVE. THE SIZE MAY DIFFER FROM RAW SIZE SIGNIFICANTLY, INDICATING DATA OR CODE MODIFICATION DURING RUNTIME.
ALSO, THE ALLOCATED SECTION SIZE IS ALWAYS PADDED TO THE SIZE MENTIONED IN SECTION_ALIGNMENT MEMBER OF THE OPTIONAL HEADER.
WHEN EXPLORING PE SECTIONS IN VIRTUAL MEMORY, BE SURE TO RECALCULATE THE OFFSETS AND SIZES OF SECTIONS, ACCORDING TO VIRTUAL SIZES AND ALIGNMENT VALUE.

WHEN EXPLORING PE SECTIONS IN VIRTUAL MEMORY, BE SURE TO RECALCULATE THE OFFSETS AND SIZES OF SECTIONS, ACCORDING TO VIRTUAL SIZES AND ALIGNMENT VALUE.

0xFF...

PE/PE32+ File Structure

ITERATING THROUGH IMPORTS AND EXPORTS:

0x70 STRUCT IMAGE_DATA_DIRECTORY[01]

0x00 DWORD IMPORT TABLES RVA

0x04 DWORD IMPORT TABLES SIZE

STRUCTURES ALLOCATED
ONE BY ONE IN .IDATA SECTION

0x00	DWORD	IMPORT NAME TABLE RVA
0x04	DWORD	TIMESTAMP
0x08	DWORD	FORWARDER CHAIN
0x0C	DWORD	DLL NAME RVA
0x10	DWORD	IMPORT ADDRESS TABLE RVA
0x14	DWORD	IMPORT NAME TABLE RVA
0x18	DWORD	TIMESTAMP
0x1C	DWORD	FORWARDER CHAIN
0x20	DWORD	DLL NAME RVA
0x24	DWORD	IMPORT ADDRESS TABLE RVA
...	...	...

POINTS DIRECTLY TO THE
DLL NAME STRING

0x00	DWORD RVA	REGOPENKEYEX
0x04	DWORD RVA	REGQUERYVALUEEX
0x08	DWORD RVA	REGCLOSEKEY
0x0C	DWORD RVA	REGCREATEKEYEXA
0x10	DWORD	0x00 0x00

EACH POINTS TO IMPORTED FUNCTION NAME STRING, ZERO ENDS ENUM

0x70 STRUCT IMAGE_DATA_DIRECTORY[00]

0x00 DWORD EXPORT TABLES RVA

0x04 DWORD EXPORT TABLES SIZE

0x00	DWORD	CHARACTERISTICS	
0x04	DWORD	TIMESTAMP	COMPILATION TIMESTAMP
0x08	WORD	MAJOR VERSION	
0x0A	WORD	MINOR VERSION	
0x0C	DWORD	NAME RVA	POINTER TO DLLNAME.DLL
0x10	DWORD	ORDINAL BASE	SUBSTRACT THIS FROM ORDINAL TO GET THE OFFSET IN TABLE
0x14	DWORD	NUMBER OF FUNCTIONS	NUMBER OF FUNCTIONS EXPORTED BY DLL
0x18	DWORD	NUMBER OF NAMES	NUMBER OF NAMED FUNCTIONS EXPORTED BY DLL
0x1C	DWORD	ADDRESS TABLE RVA	
0x20	DWORD	NAME POINTER TABLE RVA	
0x24	DWORD	ORDINAL TABLE RVA	

DO YOUR MATH WORKING ON EXPORTS. REMEMBER, ALL FUNCTIONS EXPORTED ARE PRESENT IN THE TABLES, IN THE SAME ORDER. TABLE OF NAMES HAS SMALLER NUMBER OF RECORDS THAN ORDINAL AND ADDRESS TABLE. TO MAKE POSSIBLE FINDING THE FUNCTION IN LOOP, ORDINAL BASE SHALL BE ADDEDD OR SUBSTRACTED.

TABLE OF ORDINALS.
NOT ALL FUNCTIONS ARE NAMED,
SO SOME OF THEM CAN BE RUN ONLY
USING ORDINAL. IF YOU KNOW NAME,
SUBSTRACT THE ORDINALBASE FROM
COUNTER TO FIND YOUR FUNCTION

TABLE OF POINTERS
TO FUNCTION NAME.
ALLOCATED ONE BY ONE.
NOT ALL FUNCTIONS ARE
NAMED, SO YOU MUST ADD
ORDINAL BASE TO COUNTER
TO FIND THE NAME IN ARRAY

TABLE OF POINTERS
TO FUNCTION CODE.
ALLOCATED ONE BY ONE.
ALL FUNCTIONS ARE HERE