PE File Format

SCP 20학번 박준형

목차

table of contents

1 PE Fiile 이란?

2 PE File의 기본 구조

3 PE 헤더 구조체

4 Q&A 및 피드백

PE File 이란?

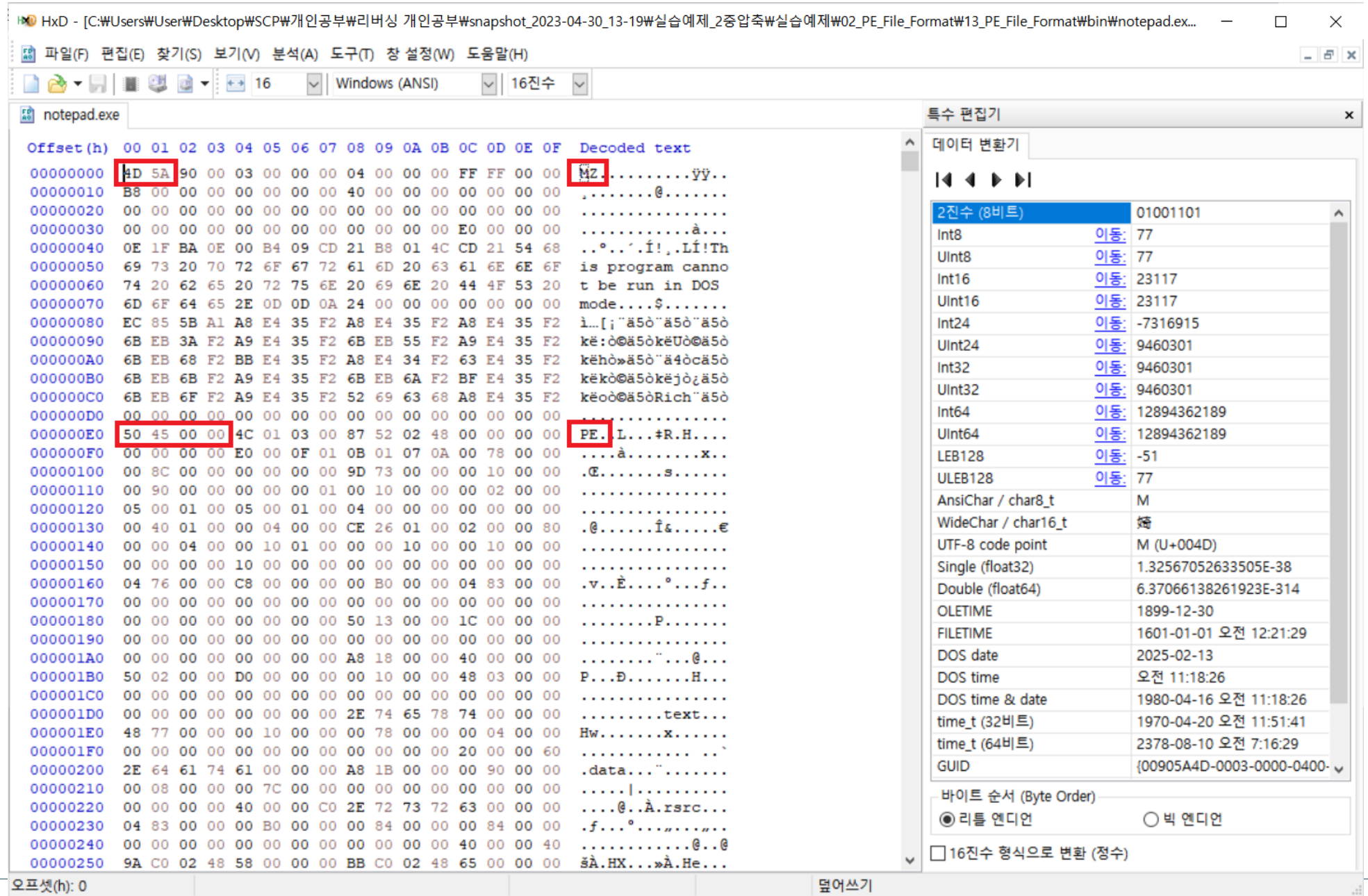
PE File 이란?

: PE(Portable Executable), Windows 운영체제에서 사용되는 실행 파일 형식

종류	주요 확장자
실행 계열	EXE, SCR
드라이버 계열	SYS, VXD
라이브러리 계열	DLL, OCX, CPL, DRV
오브젝트 파일 계열	OBJ

→ OBJ(오브젝트)파일을 제외한 모든 것은 실행 가능한 파일
(디버거, 서비스, 기타 등등의 방법을 통해)

Part 1 PE File 이란?

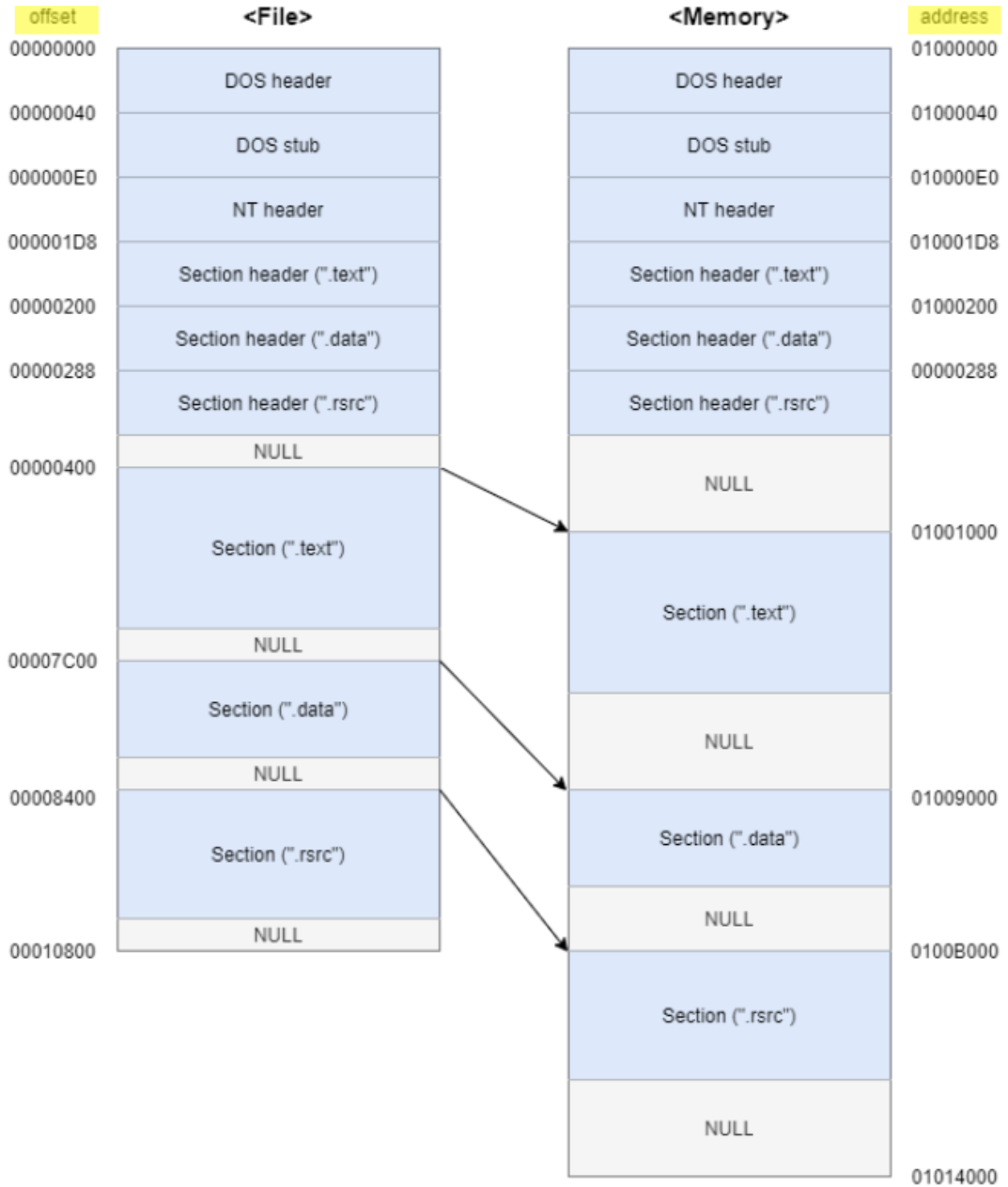


PE File의 기본 구조

Part 2 PE File의 기본 구조

PE 헤더

PE 바디



VA & RVA?

VA(Virtual Address)는 프로세스 가상 메모리의 ”절대 주소”를 말하며, RVA(Relative Virtual Address)는 어느 기준 위치(ImageBase)에서부터의 '상대 주소'를 가리킨다. 즉 다음과 같은 수식으로 나타난다.

RVA + ImageBase = VA

(상대주소 + 기준 위치 = 절대 주소)

이는 프로세스가 메모리에 로딩되는 순간 이미 다른 파일이 로딩되어 있는 경우, 재배치를 통해 다른 위치에 로딩하기 위한 것이다.

PE File 헤더 구조체

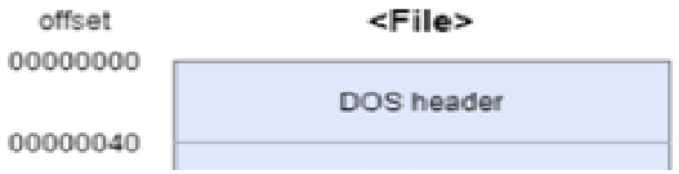
Part 3 PE 헤더 구조체 – 1) DOS Header

```
typedef struct _IMAGE_DOS_HEADER {
        WORD e_magic;    //DOS signature : 4D5A("MZ")
    WORD e_cblp;
    WORD e_cp;
    WORD e_crlc;
    WORD e_cparhdr;
    WORD e_minalloc;
    WORD e_ss;
    WORD e_sp;
    WORD e_csum;
    WORD e_ip;
    WORD e_cs;
    WORD e_lfarlc;
    WORD e_ovno;
    WORD e_res[4];
    WORD e_oemid;
    WORD e_oeminfo;
    WORD e_res2[10];
    LONG e_lfanew;      //offset to NT header
 } IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER
```

4D5A → ASCII 값 “MZ”

NT header의 옵셋을 표시

Part 3 PE 헤더 구조체– 1) DOS Header

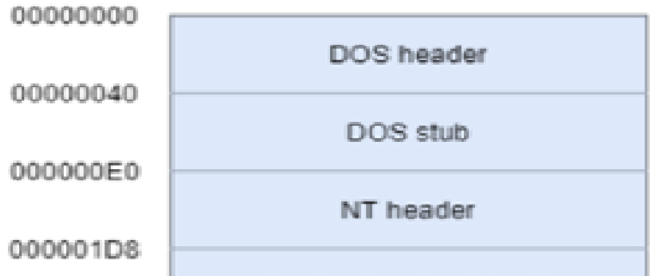


(Hex Editor)

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....ÿÿ..
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	E0	00	00	00	.....à...

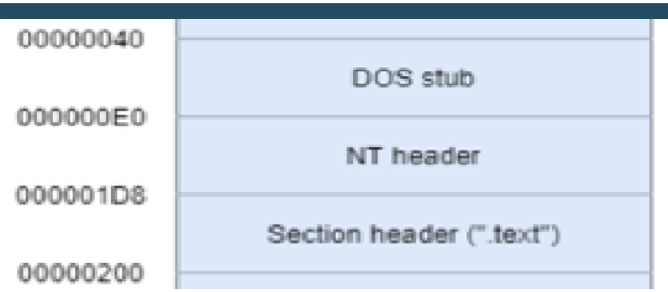
000000E0 (little endian 표기법이라 역순으로 저장)

Part 3 PE 헤더 구조체– 2) DOS Stub




```
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!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  EC 85 5B A1 A8 E4 35 F2 A8 E4 35 F2 A8 E4 35 F2  ì…[¡¨ä5ò¨ä5ò¨ä5ò
00000090  6B EB 3A F2 A9 E4 35 F2 6B EB 55 F2 A9 E4 35 F2  kë:ò©ä5òkëUò©ä5ò
000000A0  6B EB 68 F2 BB E4 35 F2 A8 E4 34 F2 63 E4 35 F2  këhò»ä5ò¨ä4òcä5ò
000000B0  6B EB 6B F2 A9 E4 35 F2 6B EB 6A F2 BF E4 35 F2  këkò©ä5òkëjò¿ä5ò
000000C0  6B EB 6F F2 A9 E4 35 F2 52 69 63 68 A8 E4 35 F2  këoò©ä5òRich¨ä5ò
000000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000E0  50 45 00 00 4C 01 03 00 87 52 02 48 00 00 00 00  PE..L...‡R.H....
000000F0  00 00 00 00 E0 00 0F 01 0B 01 07 0A 00 78 00 00  ....à........x..
```

Part 3 PE 헤더 구조체 – 3) NT Header - Signature




```
typedef struct _IMAGE_NT_HEADERS {
  DWORD                   Signature;   5045000h("PE"00)
  IMAGE_FILE_HEADER       FileHeader;
  IMAGE_OPTIONAL_HEADER32 OptionalHeader;
} IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;
```

```
000000E0  50 45 00 00 4C 01 03 00 87 52 02 48 00 00 00 00  PE..L...‡R.H....
000000F0  00 00 00 00 E0 00 0F 01 0B 01 07 0A 00 78 00 00  ....à........x..
00000100  00 8C 00 00 00 00 00 00 9D 73 00 00 00 10 00 00  .Œ.......s......
00000110  00 90 00 00 00 00 00 01 00 10 00 00 00 02 00 00  ................
00000120  05 00 01 00 05 00 01 00 04 00 00 00 00 00 00 00  ................
00000130  00 40 01 00 00 04 00 00 CE 26 01 00 02 00 00 80  .@......Î&.....€
```

Part 3 PE 헤더 구조체 – 4) NT Header - File Header

```
typedef struct _IMAGE_FILE_HEADER {
  WORD  Machine; CPU별로 고유한 값이 부여
  WORD  NumberOfSections; PE 파일에 존재하는 Section의 개수
  DWORD TimeDateStamp;
  DWORD PointerToSymbolTable;
  DWORD NumberOfSymbols;
  WORD  SizeOfOptionalHeader; IMAGE_OPTIONAL_HEADERS32 구조체의 크기
  WORD  Characteristics; 파일의 속성을 나타내는 값
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

Part 2 PE 헤더 구조체 – 5) NT Header - Optional Header (1)

```
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;
```

32bit용 구조체인 경우 10B, 64bit용 구조체인 경우 20B

EP의 RVA 값을 나타냄

PE 파일이 로딩되는 시작 주소를 나타냄

각각 메모리에서의 최소 단위, 파일에서의 최소 단위를 나타냄

PE 헤더 구조체 – 5) NT Header - Optional Header (2)

```
    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;
```

SizeOfImage: 가상 메모리에서 PE Image가 차지하는 크기를 나타냄

SizeOfHeaders: PE 헤더의 전체 크기를 나타냄

Subsystem: 시스템 드라이버인지, 일반 실행 파일인지를 구분

NumberOfRvaAndSizes: DataDirectory 배열의 개수를 나타냄

배열의 각 항목마다 정의된 값을 지님

Part 2 PE 헤더 구조체 – 6) Section Header

```
typedef struct _IMAGE_SECTION_HEADER {
  BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];
  union {
    DWORD PhysicalAddress;
    DWORD VirtualSize; 메모리에서 섹션이 차지하는 크기
  } Misc;
  DWORD VirtualAddress; 메모리에서 섹션의 시작 주소(RVA)
  DWORD SizeOfRawData; 파일에서 섹션이 차지하는 크기
  DWORD PointerToRawData;파일에서 섹션의 시작 위치
  DWORD PointerToRelocations;
  DWORD PointerToLinenumbers;
  WORD  NumberOfRelocations;
  WORD  NumberOfLinenumbers;
  DWORD Characteristics; 섹션의 속성(bit OR)
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

Q&A