Exploring the Portable Executable format

44CON

London, England

Ange Albertini

2013/09/13

Workshop package (PoCs+docs)

http://www.xchg.info/corkami/workshop.zip

Recommended PE viewer:

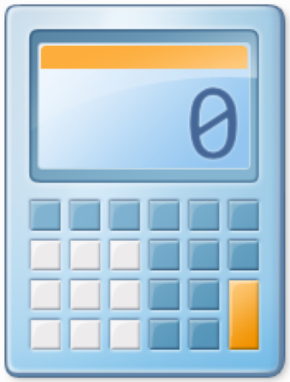
http://icerbero.com/peinsider

Portable **E**xecutable

based on

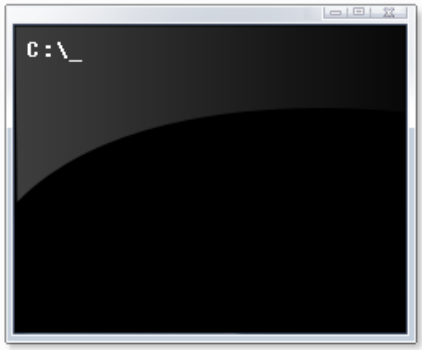
Common **O**bject **F**ile **F**ormat

AcroRd32



calc

chrome

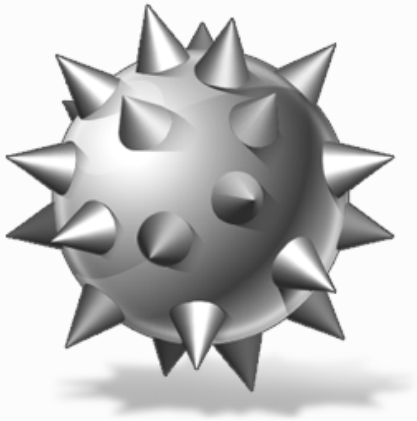


cmd

explorer

firefox

iexplore

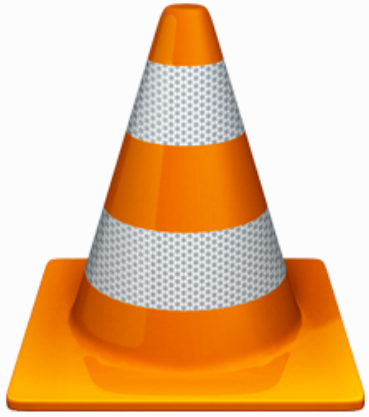


MineSweeper

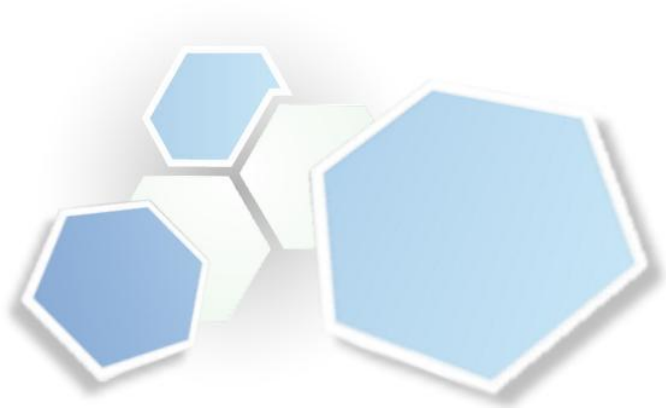
mspaint

notepad

Solitaire



vlc



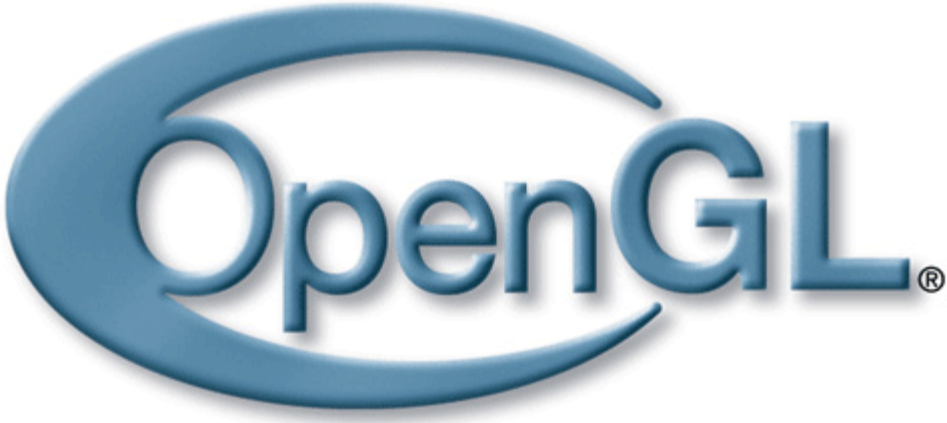

zlib


PNG

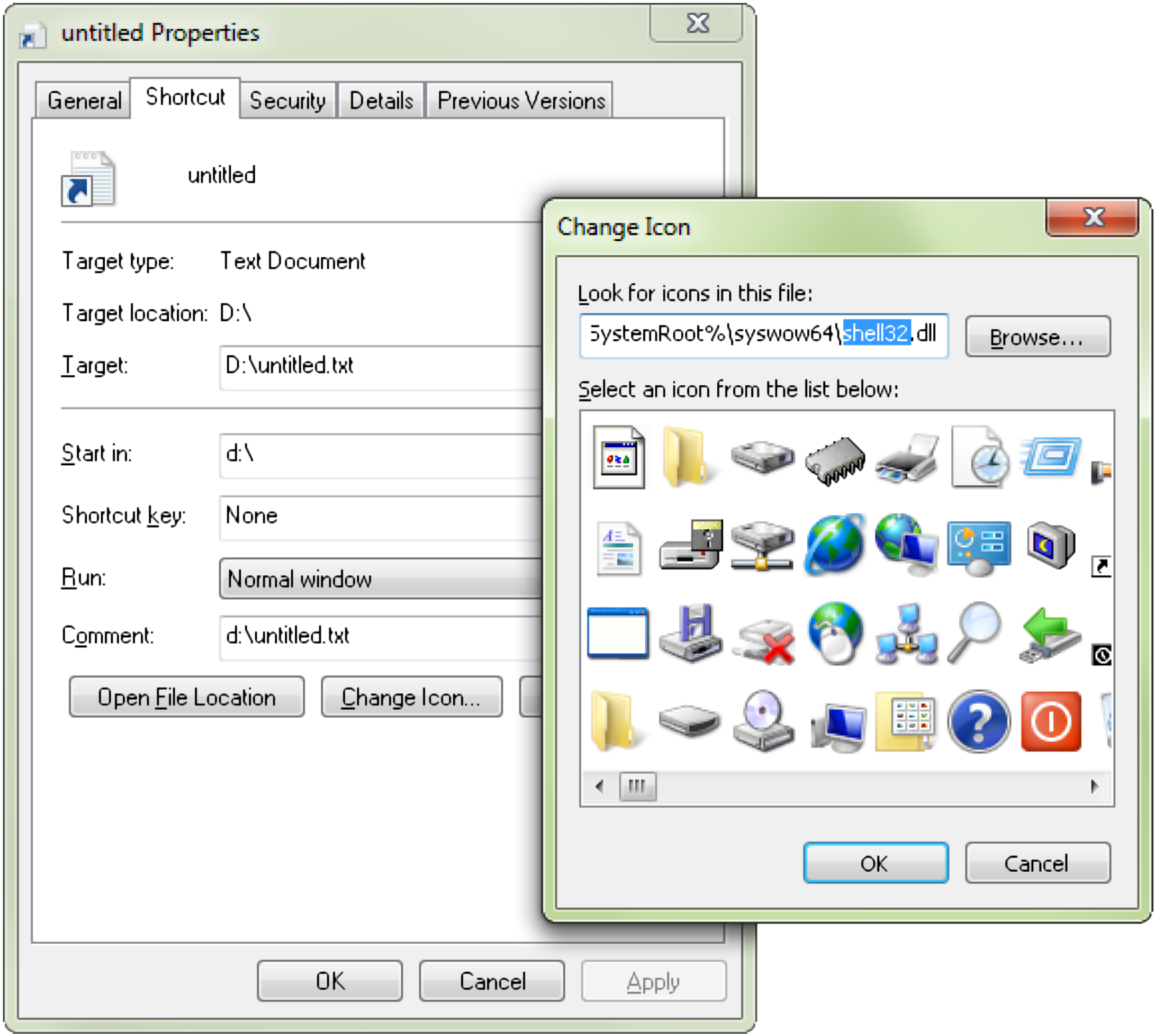

DIRECTX 11


lib
CURL


Qt


OpenGL®

- Computer
- Disk drives
- Display adapters
- DVD/CD-ROM drives
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
- Processors
- Security Devices
- Sound, video and game controllers
- System devices
- Universal Serial Bus controllers


untitled Properties
General
Shortcut
Security
Details
Previous Versions
untitled
Target type: Text Document
Target location: D:\
Target: D:\untitled.txt
Start in: d:\
Shortcut key: None
Run: Normal window
Comment: d:\untitled.txt
Open File Location
Change Icon...
OK
Cancel
Apply
Change Icon
Look for icons in this file:
%SystemRoot%\syswow64\shell32.dll
Browse...
Select an icon from the list below:
OK
Cancel

universal

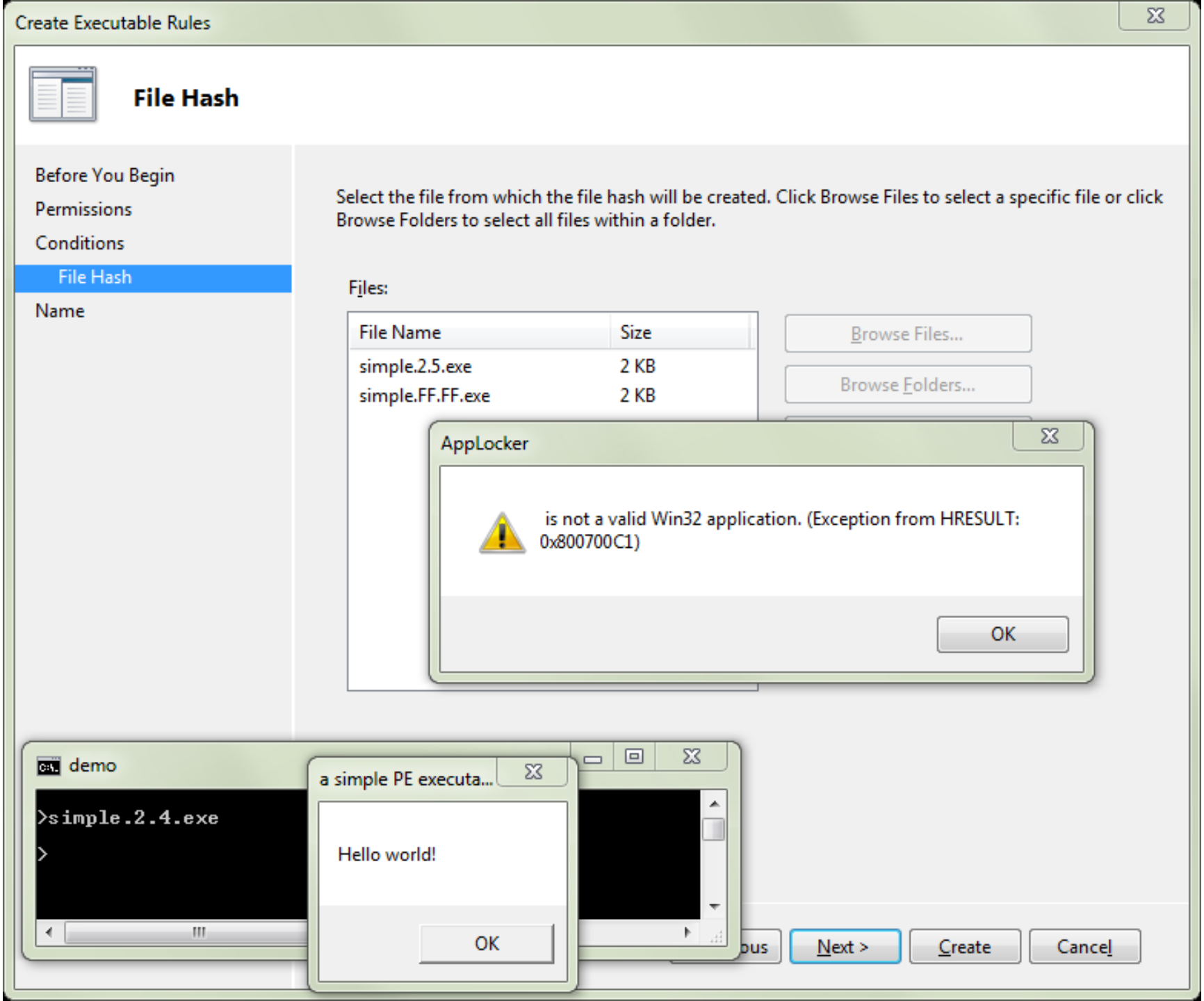
Windows binary

since 1993

Microsoft Portable Executable and Common Object File Format Specification

Revision 8.2 – September 21, 2010

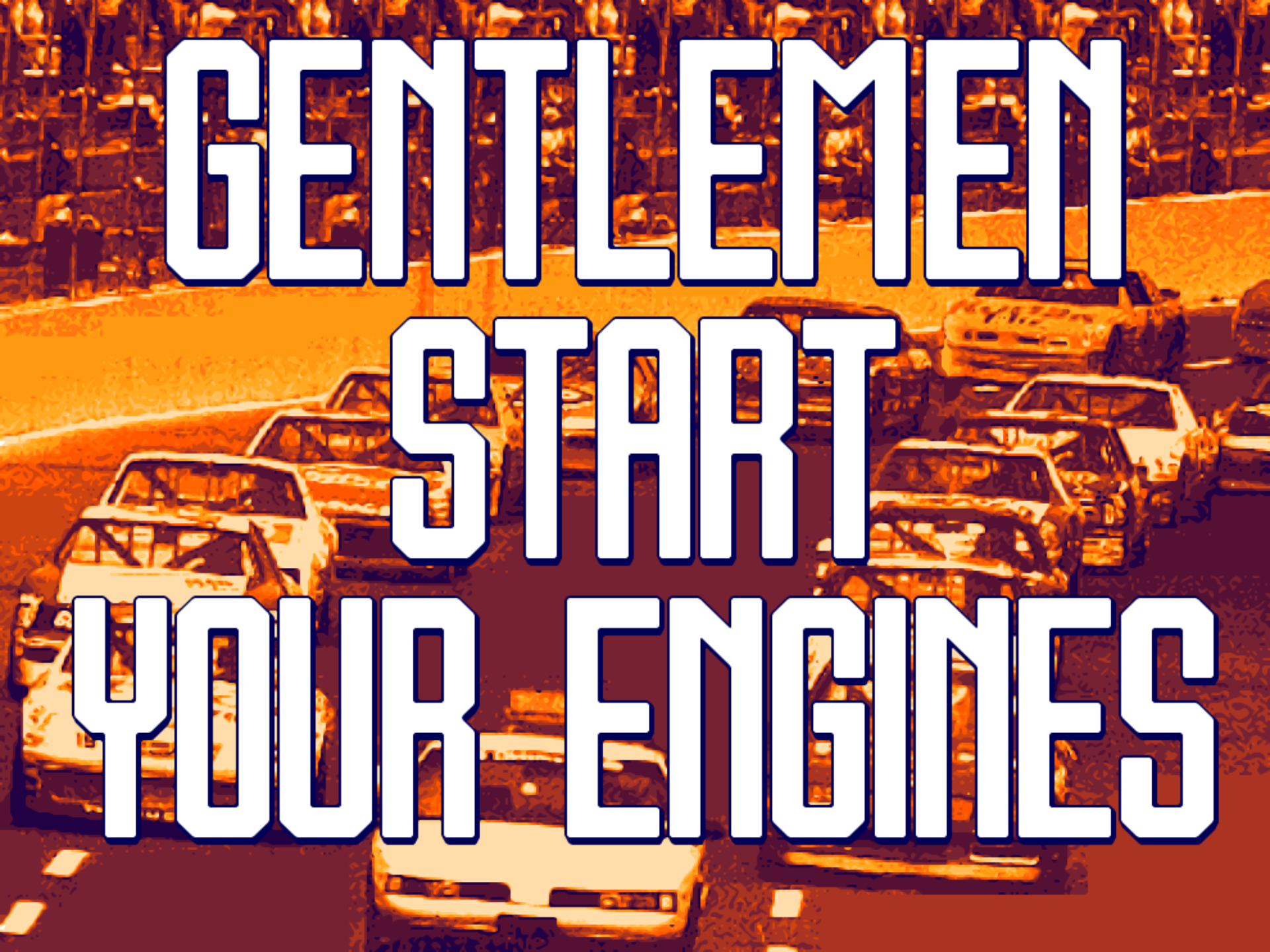
aka “the *gentle* guide to **standard** PEs”

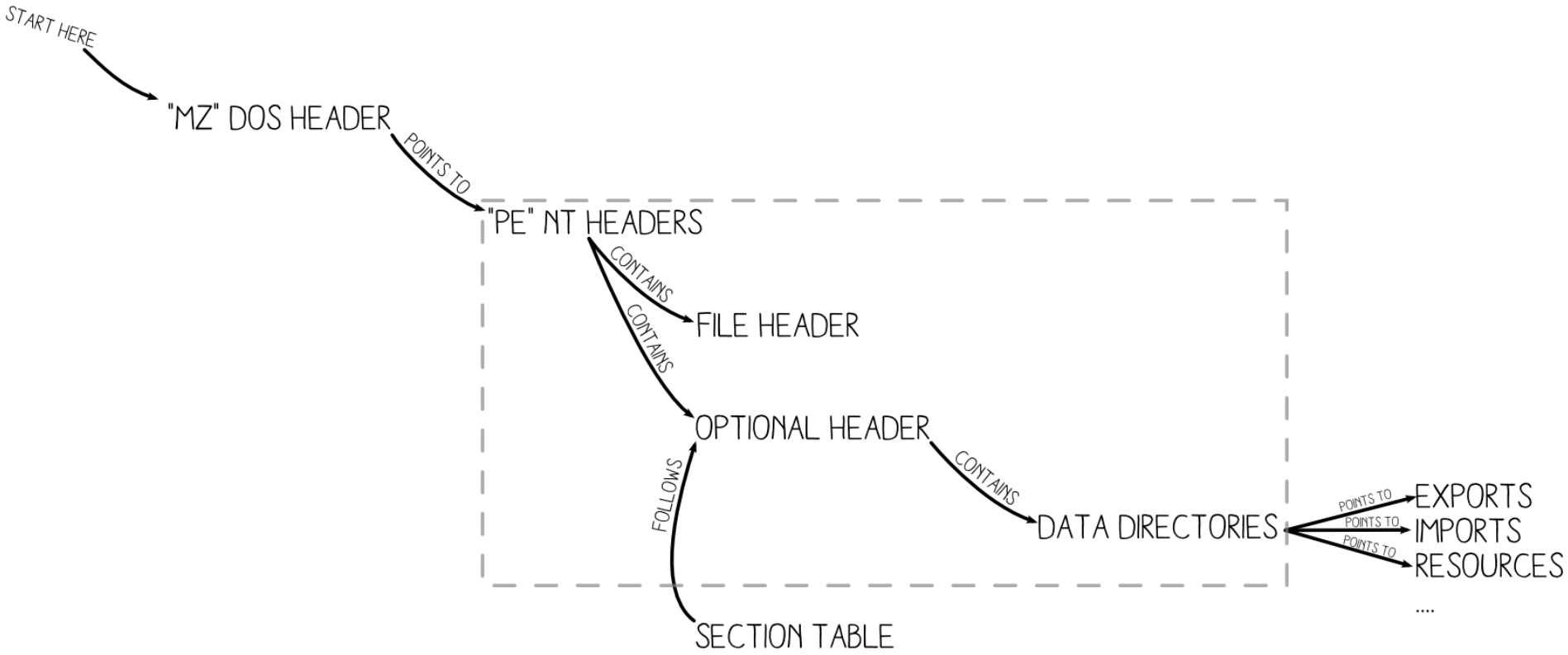

Create Executable Rules
File Hash
Before You Begin
Permissions
Conditions
File Hash
Name
Select the file from which the file hash will be created. Click Browse Files to select a specific file or click Browse Folders to select all files within a folder.
Files:
File Name
Size
simple.2.5.exe
2 KB
simple.FF.FF.exe
2 KB
Browse Files...
Browse Folders...
AppLocker
is not a valid Win32 application. (Exception from HRESULT: 0x800700C1)
OK
demo
>simple.2.4.exe
>
a simple PE executa...
Hello world!
OK
Next >
Create
Cancel


```
IMAGEBASE equ 400000h
org IMAGEBASE

istruc IMAGE_DOS_HEADER
    at IMAGE_DOS_HEADER.e_magic,  db 'MZ'
    at IMAGE_DOS_HEADER.e_lfanew, dd NT_Signature - IMAGEBASE
iend
NT_Signature:
istruc IMAGE_NT_HEADERS
    at IMAGE_NT_HEADERS.Signature, db 'PE', 0, 0
iend
istruc IMAGE_FILE_HEADER
    at IMAGE_FILE_HEADER.Machine,         dw IMAGE_FILE_MACHINE_I386
    at IMAGE_FILE_HEADER.Characteristics, dw IMAGE_FILE_EXECUTABLE_IMAGE
iend
istruc IMAGE_OPTIONAL_HEADER32
    at IMAGE_OPTIONAL_HEADER32.Magic,                  dw IMAGE_NT_OPTIONAL_HDR32_MAGIC
    at IMAGE_OPTIONAL_HEADER32.AddressOfEntryPoint,    dd EntryPoint - IMAGEBASE ; not strictly required
    at IMAGE_OPTIONAL_HEADER32.ImageBase,              dd IMAGEBASE ; not required under XP
    at IMAGE_OPTIONAL_HEADER32.SectionAlignment,       dd 1
    at IMAGE_OPTIONAL_HEADER32.FileAlignment,          dd 1
    at IMAGE_OPTIONAL_HEADER32.MajorSubsystemVersion, dw 4
    at IMAGE_OPTIONAL_HEADER32.SizeOfImage,            dd SIZEOFIMAGE
    at IMAGE_OPTIONAL_HEADER32.SizeOfHeaders,          dd SIZEOFIMAGE - 1 ; required for XP
    at IMAGE_OPTIONAL_HEADER32.Subsystem,              dw IMAGE_SUBSYSTEM_WINDOWS_CUI
iend
istruc IMAGE_DATA_DIRECTORY_16
iend

EntryPoint:
    push 42
    pop eax
    retn
```


GENTLEMEN
START
YOUR ENGINES


START HERE
"MZ" DOS HEADER
POINTS TO
"PE" NT HEADERS
CONTAINS
FILE HEADER
CONTAINS
OPTIONAL HEADER
CONTAINS
DATA DIRECTORIES
POINTS TO
EXPORTS
POINTS TO
IMPORTS
POINTS TO
RESOURCES
....
FOLLOWS
SECTION TABLE


EXE
...
CALL <DLL FUNCTION>
....
IMPORTS
DLL
...
<FUNCTION START>
....
EXPORTS

a handmade PE

simple.exe
a first real example
working ~~minimal~~

PE 101 — a windows executable walk-through

ANGE ALBERTINI
CORKAMI.COM

DISSECTED PE

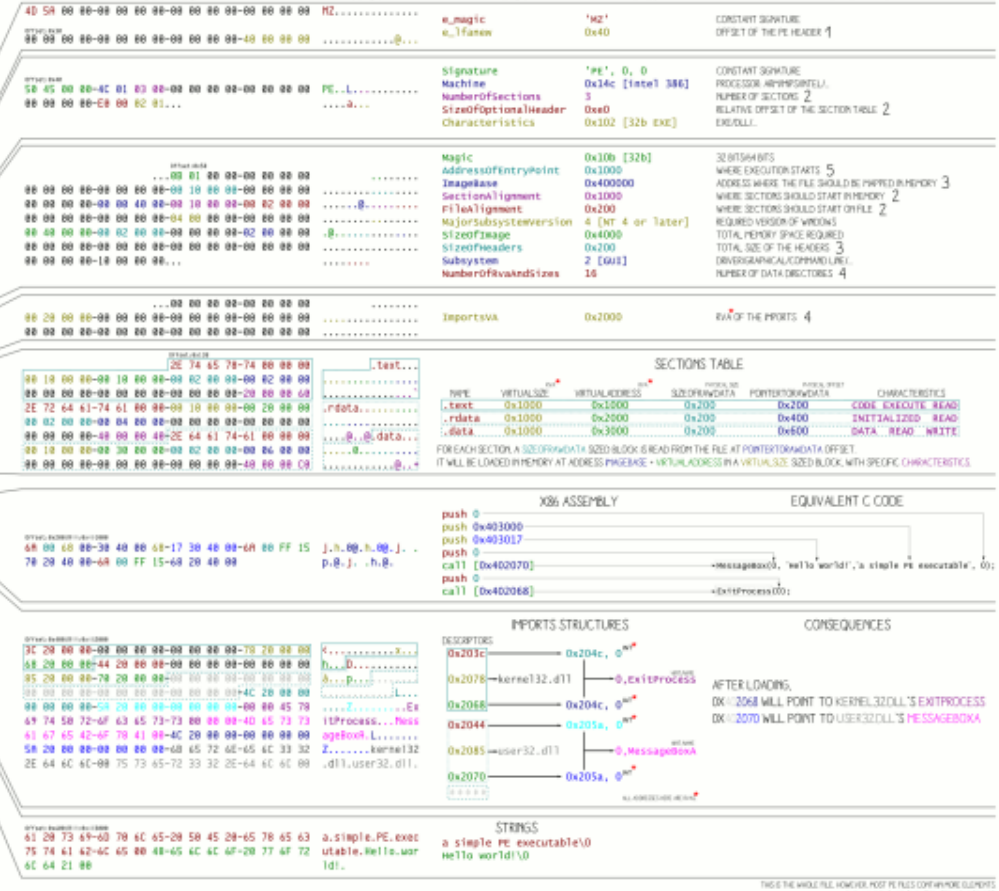
SIMPLE.EXE

HEADER
TECHNICAL DETAILS ABOUT THE EXECUTABLE

- DOS HEADER — SHOWS IT'S A WINNT
- PE HEADER — SHOWS IT'S A MODERN BINARY
- OPTIONAL HEADER — EXECUTABLE INFORMATION
- DATA DIRECTORIES — POINTERS TO EXTRA STRUCTURES (EXPORTS, IMPORTS...)
- SECTIONS TABLE — DEFINES HOW THE FILE IS LOADED IN MEMORY

SECTIONS
CONTENTS OF THE EXECUTABLE

- CODE — WHAT IS EXECUTED
- IMPORTS — LINK BETWEEN THE EXECUTABLE AND EXTERNAL LIBRARIES
- DATA — INFORMATION USED BY THE CODE

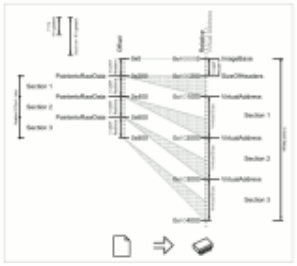


LOADING PROCESS

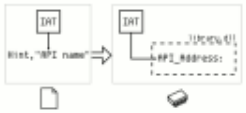
1 HEADERS
THE DOS HEADER IS PARSED
THE PE HEADER IS PARSED
ITS OFFSET IS DOS HEADER'S E_LFANEW
THE OPTIONAL HEADER IS PARSED
IT FOLLOWS THE PE HEADER

2 SECTIONS TABLE
SECTIONS TABLE IS PARSED
IT IS LOCATED AT: OFFSET OPTIONAL HEADER + SIZEOFOPTIONALHEADER
IT CONTAINS NUMBEROFSECTIONS ELEMENTS
IT IS CHECKED FOR VALIDITY WITH ALIGNMENTS
FILEALIGNMENTS AND SECTIONALIGNMENTS

3 MAPPING
THE FILE IS MAPPED IN MEMORY ACCORDING TO:
THE IMAGEBASE
THE SIZEOFHEADERS
THE SECTIONS TABLE



4 IMPORTS
DATA DIRECTORIES ARE PARSED
THEY FOLLOW THE OPTIONALHEADER
THEIR NUMBER IS NUMBEROFRVAANDSIZES
IMPORTS ARE ALWAYS #2
IMPORTS ARE PARSED
EACH DESCRIPTOR SPECIFIES A DLL NAME
THIS DLL IS LOADED IN MEMORY
IAT AND INT ARE PARSED SIMULTANEOUSLY
FOR EACH API IN INT
ITS ADDRESS IS WRITTEN IN THE IAT ENTRY



5 EXECUTION
CODE IS CALLED AT THE ENTRYPOINT
THE CALLS OF THE CODE GO VIA THE IAT TO THE APIS

NOTES

MZ HEADER AKA DOS_HEADER
STARTS WITH 'MZ' (INITIALS OF MARK ZBIKOWSKI MS-DOS DEVELOPER)

PE HEADER AKA IMAGE_FILE_HEADERS / COFF FILE HEADER
STARTS WITH 'PE' (PORTABLE EXECUTABLE)

OPTIONAL HEADER AKA IMAGE_OPTIONAL_HEADER
OPTIONAL ONLY FOR NON-STANDARD PES BUT REQUIRED FOR EXECUTABLES

RVA RELATIVE VIRTUAL ADDRESS
ADDRESS RELATIVE TO IMAGEBASE. IAT IMAGEBASE, RVA = 01
ALMOST ALL ADDRESSES OF THE HEADERS ARE RVAS
IN CODE, ADDRESSES ARE NOT RELATIVE

INT IMPORT NAME TABLE
NULL-TERMINATED LIST OF POINTERS TO HINT, NAME STRUCTURES

IAT IMPORT ADDRESS TABLE
NULL-TERMINATED LIST OF POINTERS
ON FILE IT IS A COPY OF THE INT
AFTER LOADING IT POINTS TO THE IMPORTED APIS

HINT
INDEX IN THE EXPORTS TABLE OF A DLL TO BE IMPORTED
NOT REQUIRED BUT PROVIDES A SPEED-UP BY REDUCING LOOK-UP

PE101 a Windows executable walk-through

PORTABLE EXECUTABLE

SIMPLE.EXE

SHA-1: B7AF4CB51CE38E43E0306596EB26F8FAB408CF9CB

DOWNLOAD @ PE101.CORKAMI.COM

HEADER

TECHNICAL DETAILS ABOUT THE EXECUTABLE

SECTIONS

CONTENTS OF THE EXECUTABLE

DOS HEADER

SHOWS IT'S A BINARY

PE HEADER

SHOWS IT'S A 'MODERN' BINARY

OPTIONAL HEADER

EXECUTABLE INFORMATION

DATA DIRECTORIES

POINTERS TO EXTRA STRUCTURES (EXPORTS, IMPORTS...)

SECTIONS TABLE

DEFINES HOW THE FILE IS LOADED IN MEMORY

CODE

WHAT IS EXECUTED

IMPORTS

LINK BETWEEN THE EXECUTABLE AND (WINDOWS) LIBRARIES

DATA

INFORMATION USED BY THE CODE

DISSECTED PE

HEXADECIMAL DUMP	ASCII DUMP	FIELDS	VALUES	EXPLANATION
4D 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00	MZ..............	e_magic	'MZ'	CONSTANT SIGNATURE
Offset:0x30 00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 00	............@...	e_lfanew	0x40	1 OFFSET OF THE PE HEADER
Offset:0x40 50 45 00 00-4C 01 03 00-00 00 00 00-00 00 00 00	PE..L...........	Signature	'PE', 0, 0	CONSTANT SIGNATURE
		Machine	0x14c [intel 386]	PROCESSOR: ARM/MIPS/INTEL/...
		NumberOfSections	3	2 NUMBER OF SECTIONS
00 00 00 00-E0 00 02 01...	....a...	SizeOfOptionalHeader	0xe0	2 RELATIVE OFFSET OF THE SECTION TABLE
		Characteristics	0x102 [32b EXE]	EXE/DLL/...
Offset:0x58 ...0B 01 00 00-00 00 00 00	....	Magic	0x10b [32b]	32 BITS/64 BITS
00 00 00 00-00 00 00 00-00 10 00 00-00 00 00 00	....	AddressOfEntryPoint	0x1000	5 WHERE EXECUTION STARTS
00 00 00 00-00 00 40 00-00 10 00 00-00 02 00 00	......@.........	ImageBase	0x400000	3 ADDRESS WHERE THE FILE SHOULD BE MAPPED IN MEMORY
00 00 00 00-00 00 00 00-04 00 00 00-00 00 00 00	....	SectionAlignment	0x1000	2 WHERE SECTIONS SHOULD START IN MEMORY
00 40 00 00-00 02 00 00-00 00 00 00-02 00 00 00	.@..............	FileAlignment	0x200	2 WHERE SECTIONS SHOULD START ON FILE
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	....	MajorSubsystemVersion	4 [NT 4 or later]	REQUIRED VERSION OF WINDOWS
00 00 00 00-10 00 00 00...	....	SizeOfImage	0x4000	TOTAL MEMORY SPACE REQUIRED
		SizeOfHeaders	0x200	3 TOTAL SIZE OF THE HEADERS
		Subsystem	2 [GUI]	DRIVER/GRAPHICAL/COMMAND LINE/...
		NumberOfRvaAndSizes	16	4 NUMBER OF DATA DIRECTORIES
...00 00 00 00-00 00 00 00	....			
00 20 00 00-00 00 00 00-00 00 00 00-00 00 00 00	....	ImportsVA	0x2000	4 RVA* OF THE IMPORTS
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	....			

Offset:0x138

2E 74 65 78-74 00 00 00	.text...
00 10 00 00-00 10 00 00-00 02 00 00-00 02 00 00	....
00 00 00 00-00 00 00 00-00 00 00 00-20 00 00 60	....`
2E 72 64 61-74 61 00 00-00 10 00 00-00 20 00 00	.rdata..... ....
00 02 00 00-00 04 00 00-00 00 00 00-00 00 00 00	....
00 00 00 00-40 00 00 40-2E 64 61 74-61 00 00 00	....@..@.data...
00 10 00 00-00 30 00 00-00 02 00 00-00 06 00 00	.....0..........
00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 C0	............@..+

SECTIONS TABLE

NAME	VIRTUALSIZE (RVA*)	VIRTUALADDRESS (RVA*)	SIZEOFRAWDATA (PHYSICAL SIZE)	POINTERTORAWDATA (PHYSICAL OFFSET)	CHARACTERISTICS
.text	0x1000	0x1000	0x200	0x200	CODE EXECUTE READ
.rdata	0x1000	0x2000	0x200	0x400	INITIALIZED READ
.data	0x1000	0x3000	0x200	0x600	DATA READ WRITE

FOR EACH SECTION, A SIZEOFRAWDATA SIZED BLOCK IS READ FROM THE FILE AT POINTERTORAWDATA OFFSET.
IT WILL BE LOADED IN MEMORY AT ADDRESS IMAGEBASE + VIRTUALADDRESS IN A VIRTUALSIZE SIZED BLOCK, WITH SPECIFIC CHARACTERISTICS.

Offset:0x200/RVA:0x401000

6A 00 68 00-30 40 00 68-17 30 40 00-6A 00 FF 15	j.h.0@.h.0@.j. .
70 20 40 00-6A 00 FF 15-68 20 40 00	p.@.j. .h.@.

X86 ASSEMBLY

```
push 0
push 0x403000
push 0x403017
push 0
call [0x402070]
push 0
call [0x402068]
```

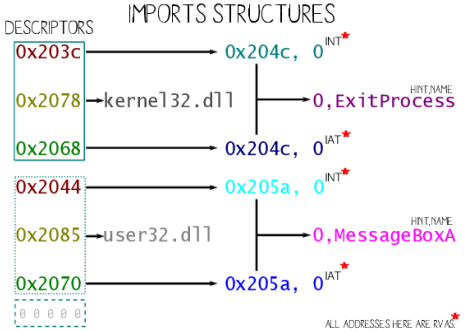
EQUIVALENT C CODE

```
MessageBox(0, "Hello world!","a simple PE executable", 0);
ExitProcess(0);
```

Offset:0x400/RVA:0x402000

3C 20 00 00-00 00 00 00-00 00 00 00-78 20 00 00	<...........x...
68 20 00 00-44 20 00 00-00 00 00 00-00 00 00 00	h...D...........
85 20 00 00-70 20 00 00-00 00 00 00-00 00 00 00	à...p...3.......
00 00 00 00-00 00 00 00-00 00 00 00-4C 20 00 00	............L...
00 00 00 00-5A 20 00 00-00 00 00 00-00 00 45 78	....Z.........Ex
69 74 50 72-6F 63 65 73-73 00 00 00-4D 65 73 73	itProcess...Mess
61 67 65 42-6F 78 41 00-4C 20 00 00-00 00 00 00	ageBoxA.L.......
5A 20 00 00-00 00 00 00-6B 65 72 6E-65 6C 33 32	Z.......kernel32
2E 64 6C 6C-00 75 73 65-72 33 32 2E-64 6C 6C 00	.dll.user32.dll.

IMPORTS STRUCTURES



CONSEQUENCES

AFTER LOADING,
0X402068 WILL POINT TO KERNEL32.DLL`S EXITPROCESS
0X402070 WILL POINT TO USER32.DLL`S MESSAGEBOXA

Offset:0x600/RVA:0x403000

61 20 73 69-6D 70 6C 65-20 50 45 20-65 78 65 63	a.simple.PE.exec
75 74 61 62-6C 65 00 48-65 6C 6C 6F-20 77 6F 72	utable.Hello.wor
6C 64 21 00	ld!.

STRINGS

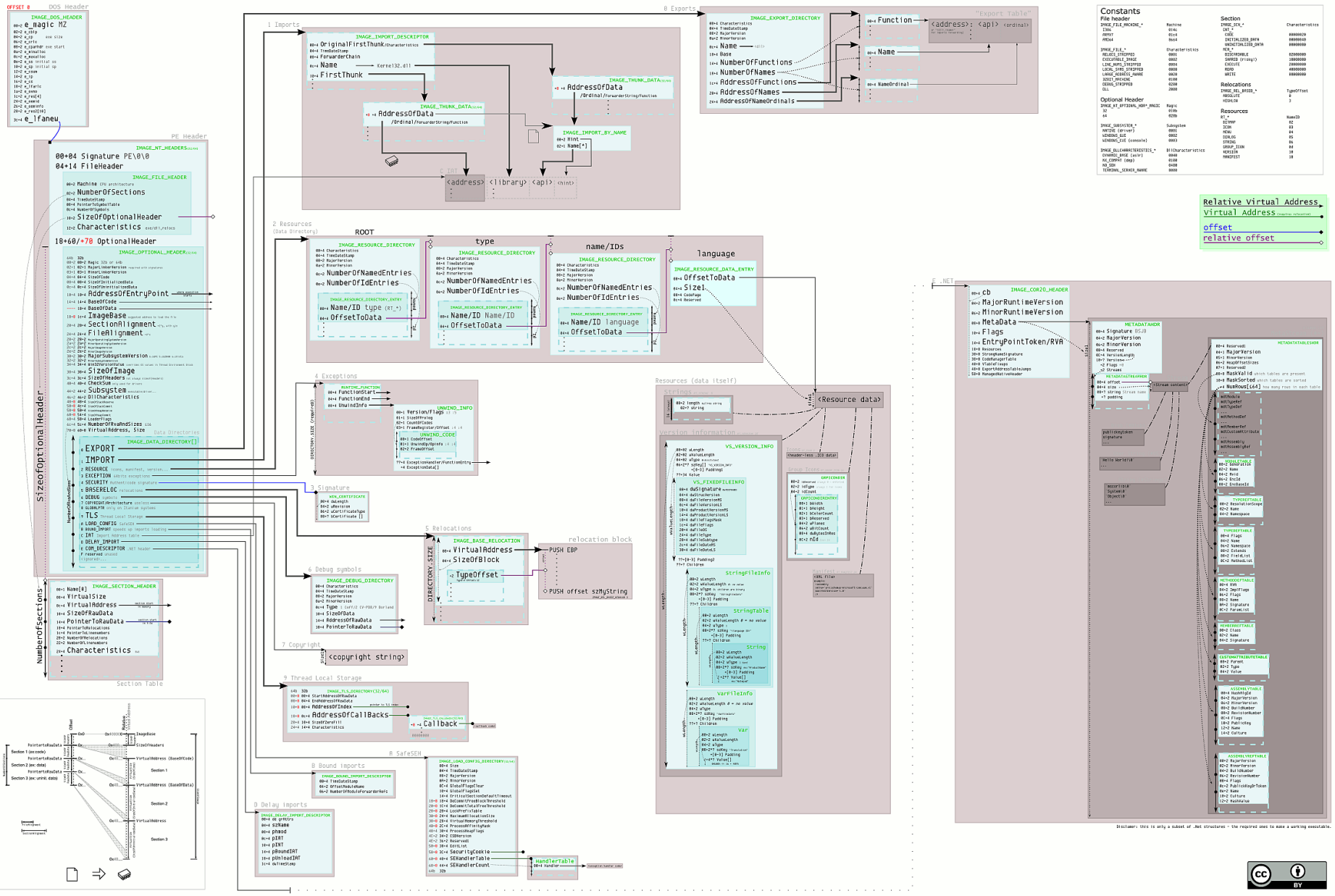
```
a simple PE executable\0
Hello world!\0
```

THIS IS THE WHOLE FILE, HOWEVER, MOST PE FILES CONTAIN MORE ELEMENTS.
EXPLANATIONS ARE SIMPLIFIED, FOR CONCISENESS.

detailed walkthrough

PE102 a Windows executable format overview

Ange Albertini
2009-2013 Corkami



DOS header

unused in PE mode

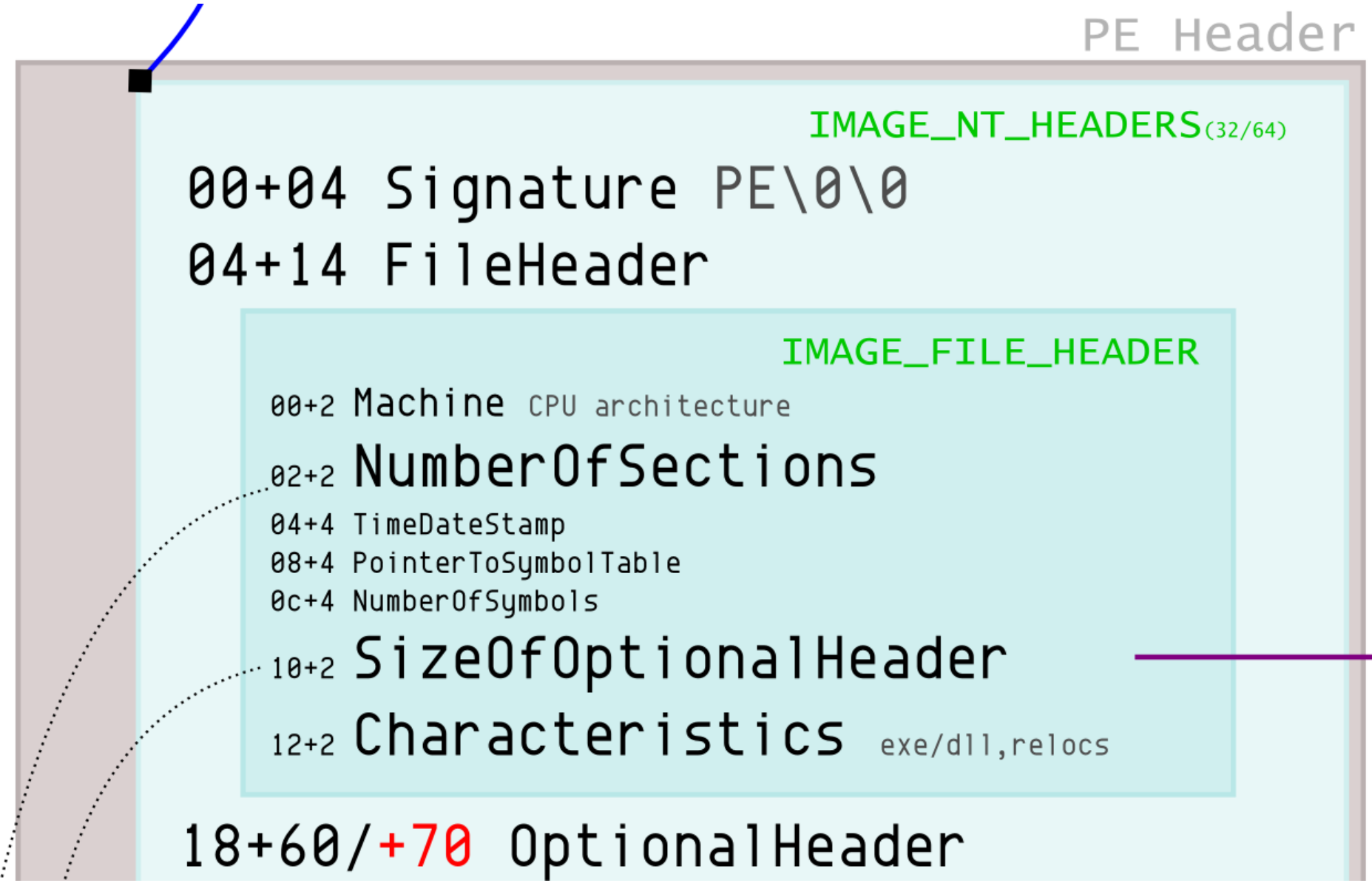
OFFSET 0 DOS Header

IMAGE_DOS_HEADER

00+2 e_magic MZ
02+2 e_cblp
04+2 e_cp exe size
06+2 e_crlc
08+2 e_cparhdr exe start
0a+2 e_minalloc
0c+2 e_maxalloc
0e+2 e_ss initial ss
10+2 e_sp initial sp
12+2 e_csum
14+2 e_ip
16+2 e_cs
18+2 e_lfarlc
1a+2 e_ovno
1c+2 e_res[4]
24+2 e_oemid
26+2 e_oeminfo
28+2 e_res2[10]
3c+4 e_lfanew

PE header

PE signature


PE Header
IMAGE_NT_HEADERS(32/64)
00+04 Signature PE\0\0
04+14 FileHeader
IMAGE_FILE_HEADER
00+2 Machine CPU architecture
02+2 NumberOfSections
04+4 TimeDateStamp
08+4 PointerToSymbolTable
0c+4 NumberOfSymbols
10+2 SizeOfOptionalHeader
12+2 Characteristics exe/dll,relocs
18+60/+70 OptionalHeader

Optional Header

NOT optional in executables

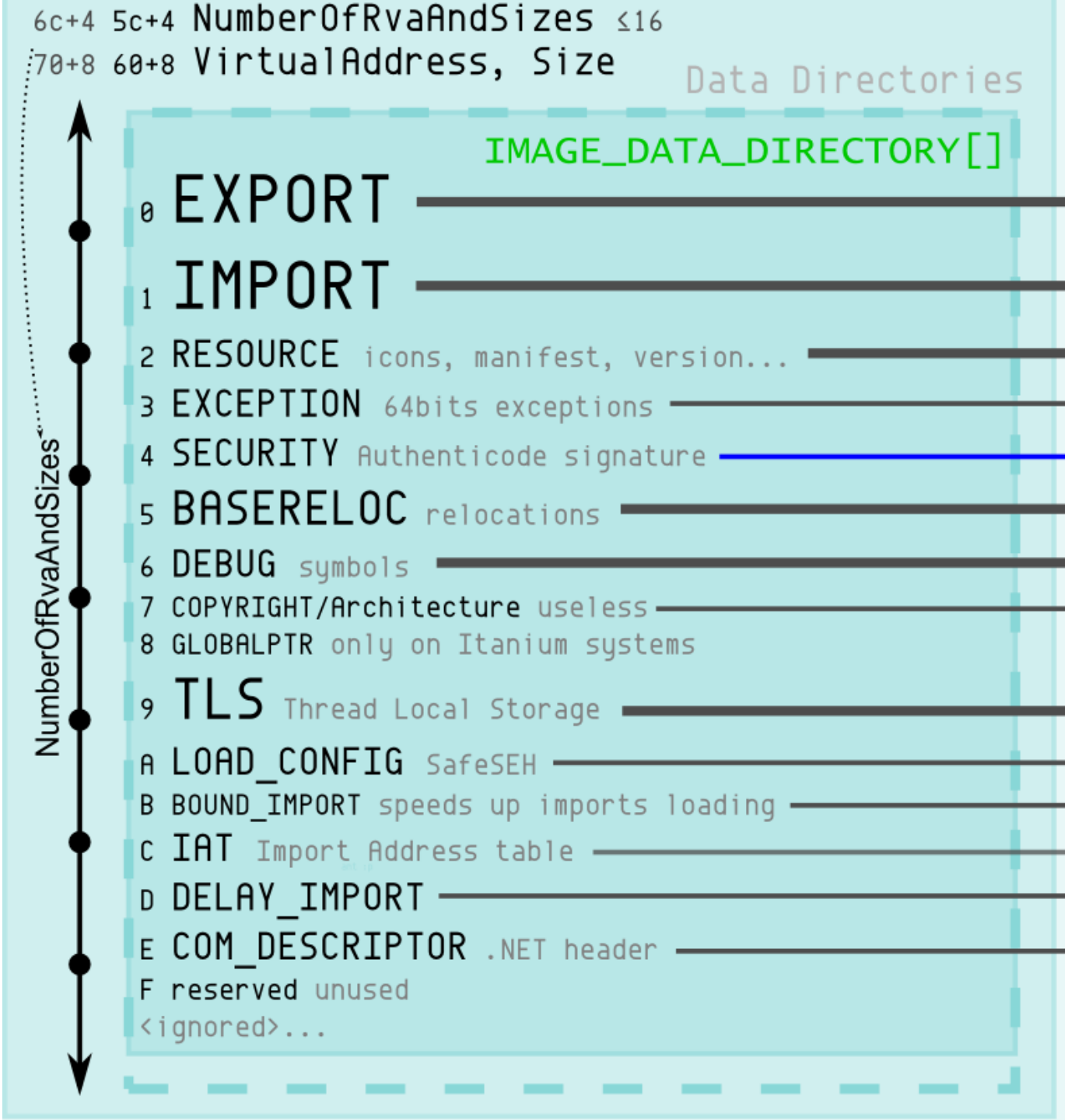
18+60/+70 OptionalHeader

IMAGE_OPTIONAL_HEADER(32/64)

64b	32b	
00+2	00+2	Magic 32b or 64b
02+1	02+1	MajorLinkerVersion required with signatures
03+1	03+1	MinorLinkerVersion
04+4	04+4	SizeOfCode
08+4	08+4	SizeOfInitializedData
0c+4	0c+4	SizeOfUninitializedData
10+4	10+4	**AddressOfEntryPoint** → where execution starts
14+4	14+4	**BaseOfCode** →
----	18+4	**BaseOfData** →
18+8	1c+4	**ImageBase** suggested address to load the file
20+4	20+4	**SectionAlignment** =2^y, with y≥x
24+4	24+4	**FileAlignment** =2^x
28+2	28+2	MajorOperatingSystemVersion
2a+2	2a+2	MinorOperatingSystemVersion
2c+2	2c+2	MajorImageVersion
2e+2	2e+2	MinorImageVersion
30+2	30+2	**MajorSubsystemVersion** 4:≥W95 5:≥W2000 6:≥Vista
32+2	32+2	MinorSubsystemVersion
34+4	34+4	Win32VersionValue overrides OS values in Thread Environment Block
38+4	38+4	**SizeOfImage**
3c+4	3c+4	**SizeOfHeaders** not always sizeof(Headers)
40+4	40+4	**CheckSum** only used for drivers
44+2	44+2	**Subsystem** executable/driver...
46+2	46+2	**DllCharacteristics**
48+8	48+4	SizeOfStackReserve
50+8	4c+4	SizeOfStackCommit
58+8	50+4	SizeOfHeapReserve
60+8	54+4	SizeOfHeapCommit
68+4	58+4	LoaderFlags
6c+4	5c+4	**NumberOfRvaAndSizes** ≤16
70+8	60+8	**VirtualAddress, Size** Data Directories

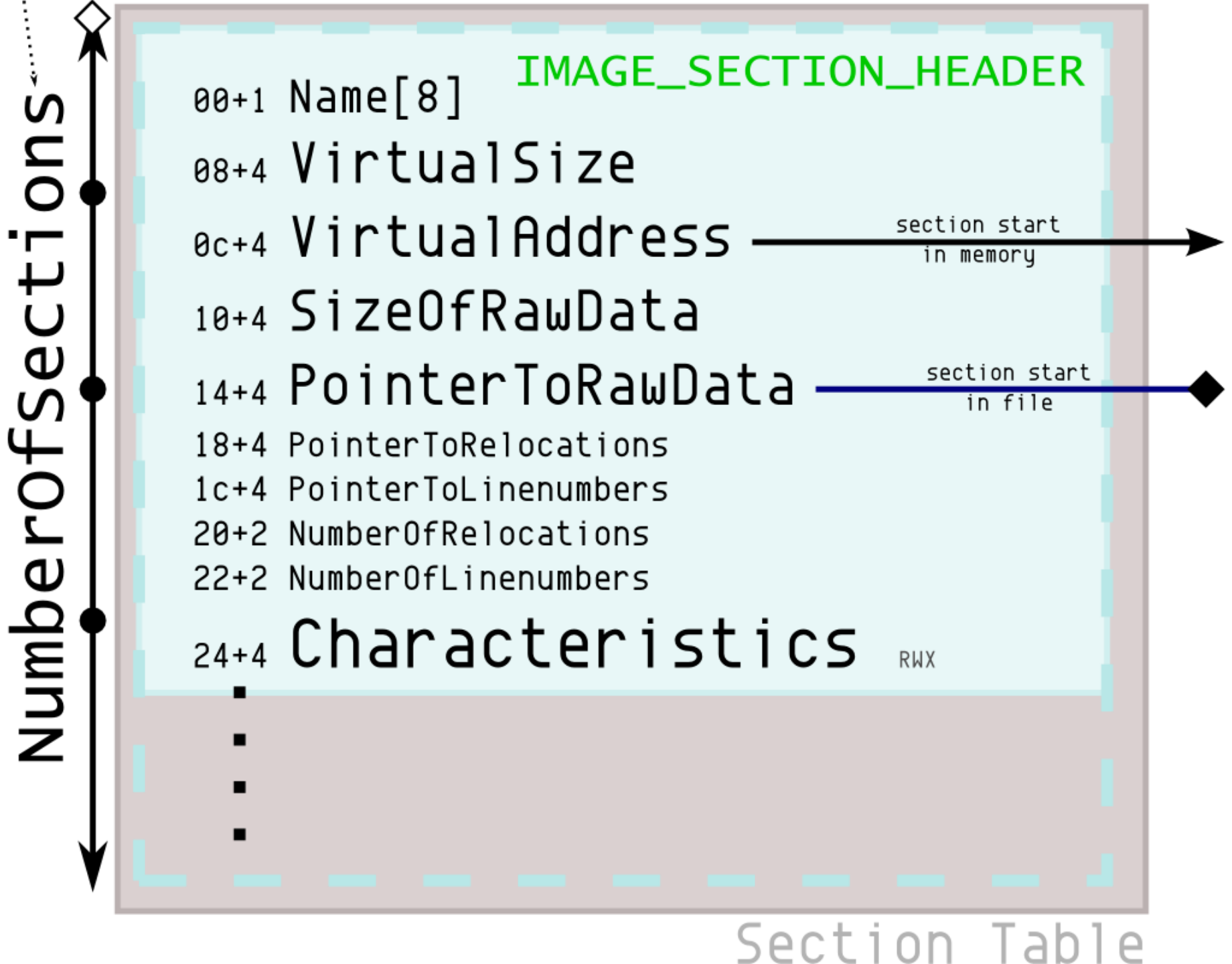
DataDirectories

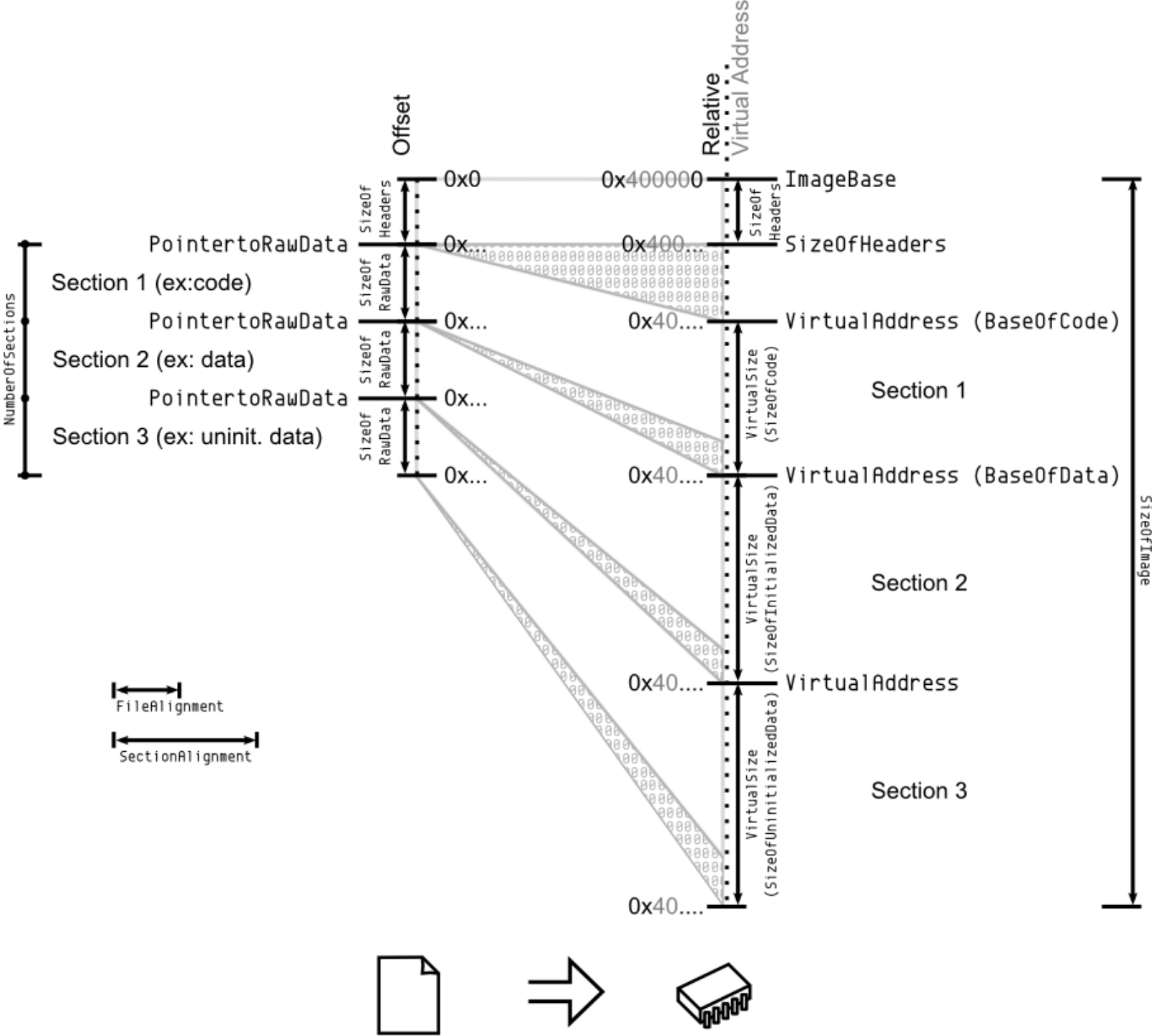
end of OptionalHeader
16 (max) * [RVA, Size]
each entry interpreted differently


6c+4 5c+4 NumberOfRvaAndSizes ≤16
70+8 60+8 VirtualAddress, Size
Data Directories
IMAGE_DATA_DIRECTORY[]
NumberOfRvaAndSizes
0 EXPORT
1 IMPORT
2 RESOURCE icons, manifest, version...
3 EXCEPTION 64bits exceptions
4 SECURITY Authenticode signature
5 BASERELOC relocations
6 DEBUG symbols
7 COPYRIGHT/Architecture useless
8 GLOBALPTR only on Itanium systems
9 TLS Thread Local Storage
A LOAD_CONFIG SafeSEH
B BOUND_IMPORT speeds up imports loading
C IAT Import Address table
D DELAY_IMPORT
E COM_DESCRIPTOR .NET header
F reserved unused
<ignored>...

Sections

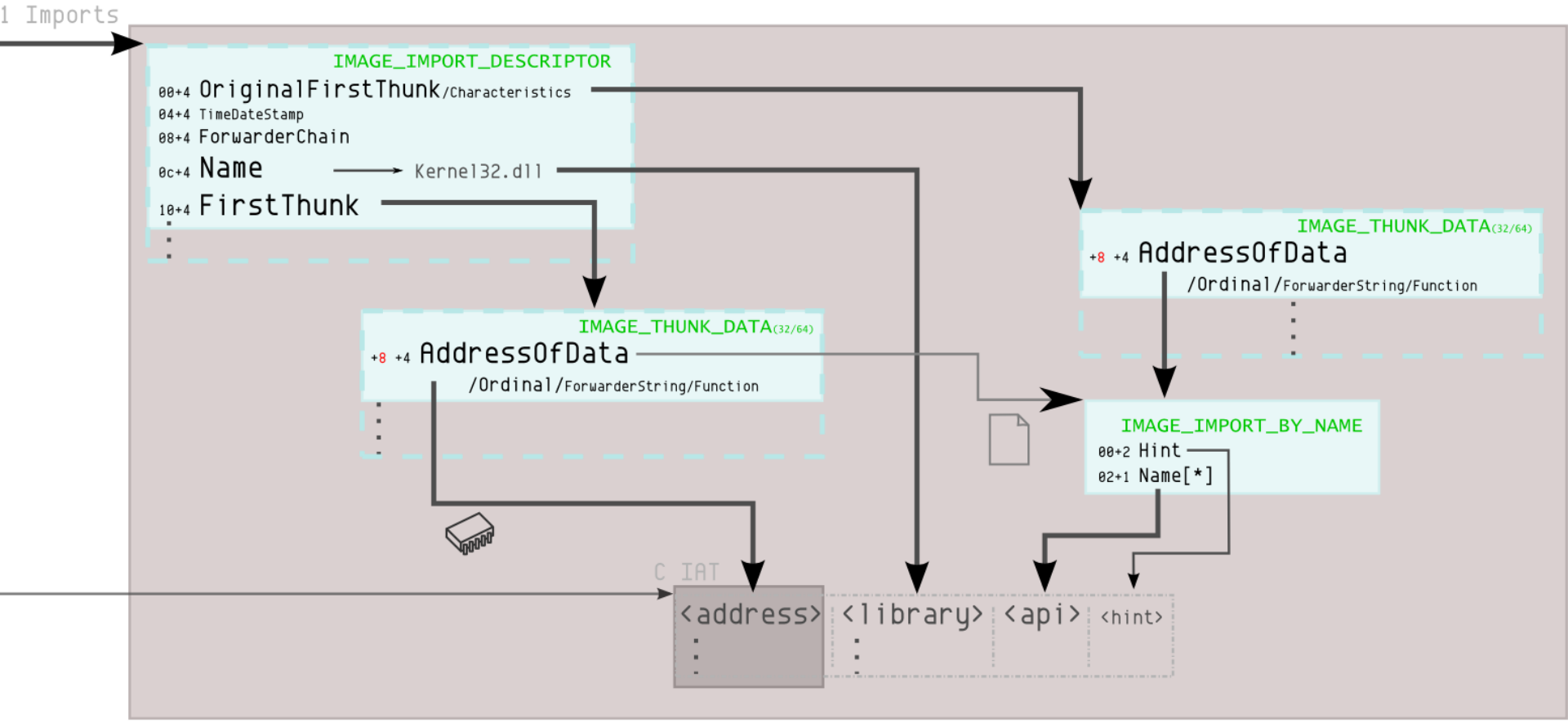
memory mapping


NumberOfSections
IMAGE_SECTION_HEADER
00+1 Name[8]
08+4 VirtualSize
0c+4 VirtualAddress
10+4 SizeOfRawData
14+4 PointerToRawData
18+4 PointerToRelocations
1c+4 PointerToLinenumbers
20+2 NumberOfRelocations
22+2 NumberOfLinenumbers
24+4 Characteristics
RWX
section start in memory
section start in file
Section Table


Offset
Relative Virtual Address
0x0
0x400000
ImageBase
SizeOf Headers
PointertoRawData
0x...
0x400...
SizeOfHeaders
Section 1 (ex:code)
SizeOf RawData
PointertoRawData
0x...
0x40....
VirtualAddress (BaseOfCode)
NumberOfSections
Section 2 (ex: data)
PointertoRawData
0x...
VirtualSize (SizeOfCode)
Section 1
Section 3 (ex: uninit. data)
0x...
0x40....
VirtualAddress (BaseOfData)
VirtualSize (SizeOfInitializedData)
Section 2
0x40....
VirtualAddress
FileAlignment
SectionAlignment
VirtualSize (SizeOfUninitializedData)
Section 3
0x40....
SizeOfImage

Imports

standard loader mechanism
NOT required
load DLL, locate APIs


1 Imports
IMAGE_IMPORT_DESCRIPTOR
00+4 OriginalFirstThunk/Characteristics
04+4 TimeDateStamp
08+4 ForwarderChain
0c+4 Name
Kernel32.dll
10+4 FirstThunk
IMAGE_THUNK_DATA(32/64)
+8 +4 AddressOfData
/Ordinal/ForwarderString/Function
IMAGE_THUNK_DATA(32/64)
+8 +4 AddressOfData
/Ordinal/ForwarderString/Function
IMAGE_IMPORT_BY_NAME
00+2 Hint
02+1 Name[*]
C IAT
<address>
<library>
<api>
<hint>

compiled PE

compiled.exe
closer to reality
extra non-critical structure

DOS Stub

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	♬ º♬.´.Í!¸ºLÍ!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....$.......

```
start:
    push    cs
    pop     ds
    mov     dx, msg - start
    mov     ah, 9
    int     21h     ; print string

    mov     ax, 4c01h
    int     21h     ; exit
msg:
    db 'This program cannot be run in DOS mode.', 0dh, 0dh, 0ah, '$'
```

- obsolete 16b code
  - prints msg & exits
- still present on all standard PEs
  - even 64b binaries

COM 101 — a DOS executable walk-through

MAND FILE

VERSION 1.0
26 MARCH 2013

ANGE ALBERTINI
CORKAMI.COM

DISSECTED FILE

SHA-1: 396E4D1776B1DDE6F5C78C14FE285ZDC53CB80A3
DOWNLOAD @ PE101.CORKAMI.COM

```
C:\>SIMPLE.COM
Hello World!
```

```
0E 1F BA 0E 01 B4 09 CD 21 B8 01 4C CD 21 48 65 ........!..L.!He
6C 6C 6F 20 57 6F 72 6C 64 21 0D 0D 0A 24       llo.World!...$
```

SIMPLE.COM

CODE — WHAT'S EXECUTED

```
Offset:    0000
Address: cs:0100
0E 1F BA 0E 01 B4 09 CD 21 B8 01 4C CD 21 . . .  ........!..L.!..
```

X86 16BITS ASSEMBLY*

```
push CS
pop  DS

1 mov  DX, msg
  mov  AH, 9
  int  0x21

2 mov  AX, 0x4C01
  int  0x21
```

EQUIVALENT C CODE

```
// copy CODE segment to DATA segment

print("Hello World!\r\r\n");

return 1;
```

*THE DOS CODE OF A STANDARD PORTABLE EXECUTABLE FILE IS IDENTICAL

DATA — INFORMATION USED BY THE CODE

```
Offset:    000E
Address: cs:010E
. . . . . . . . . . . . . . . . . . 48 65  ..............He
6C 6C 6F 20 57 6F 72 6C 64 21 0D 0D 0A 24  llo.World!...$
```

STRING

```
"Hello World!\r\r\n" $-terminated
```

EXPLANATIONS ARE SIMPLIFIED, FOR CONCISENESS

TRIVIA

COM FILES ARE USED BY MICROSOFT SINCE 1981
AND STILL RUN ON ANY 32-BIT WINDOWS TODAY

KEY CONCEPTS

FILE SPECIFICATIONS

NO HEADER
LIMITED TO ≈64KiB
MERGED CODE AND DATA
USUALLY WITH NO BOUNDARIES

MEMORY SEGMENTATION

DOS MEMORY IS CUT INTO BLOCKS
THEY ARE CALLED SEGMENTS
ADDRESSES ARE DEFINED BY:
A SEGMENT
AN OFFSET WITHIN THAT SEGMENT
THEY ARE WRITTEN `segment:offset`

LOADING PROCESS

THE FILE IS ENTIRELY LOADED
OFFSET `0` IS MAPPED AT `cs:0x100`
EXECUTION STARTS THERE

OPERATING SYSTEM'S FUNCTIONS

CALLED VIA INTERRUPTS + FUNCTION CODE IN `AH`
EXAMPLES:

1. INTERRUPT `0x21` FUNCTION `0x09`
   PRINTS A $-TERMINATED STRING AT `DS:DX`
2. INTERRUPT `0x21` FUNCTION `0x4C`
   TERMINATES THE PROCESS
   RETURN VALUE GIVEN IN `AL`

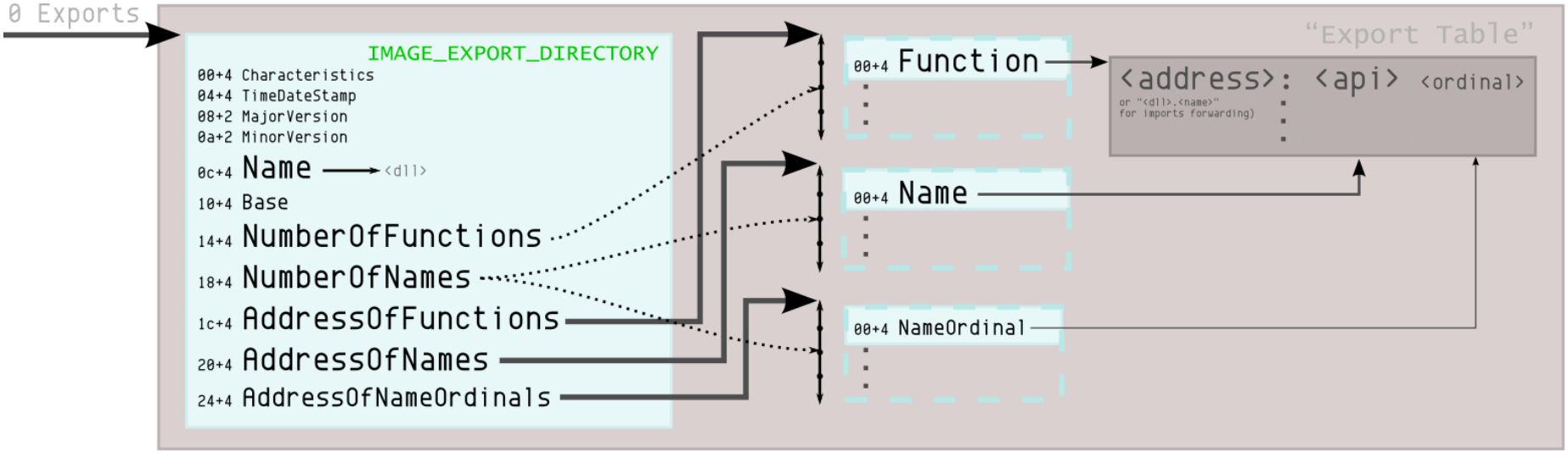
'Rich' header

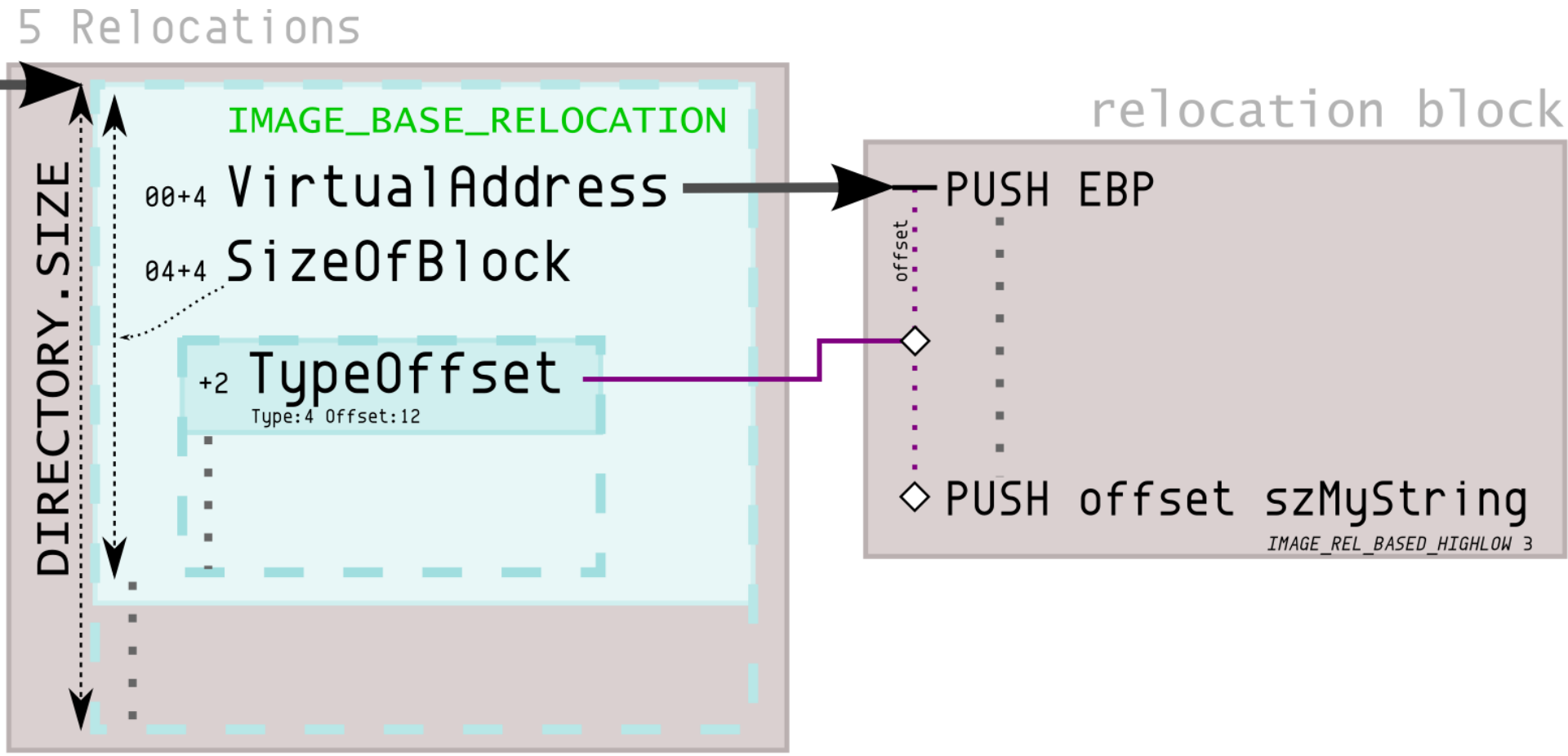
- compiler information
- officially undocumented
  - pitiful xor32 encryption
- completely documented by Daniel Pistelli
  http://ntcore.com/files/richsign.htm

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
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	....

DLL

exports
relocations


0 Exports
IMAGE_EXPORT_DIRECTORY
00+4 Characteristics
04+4 TimeDateStamp
08+2 MajorVersion
0a+2 MinorVersion
0c+4 Name
<dll>
10+4 Base
14+4 NumberOfFunctions
18+4 NumberOfNames
1c+4 AddressOfFunctions
20+4 AddressOfNames
24+4 AddressOfNameOrdinals
00+4 Function
00+4 Name
00+4 NameOrdinal
"Export Table"
<address>: <api> <ordinal>
or "<dll>.<name>"
for imports forwarding)


5 Relocations
DIRECTORY.SIZE
IMAGE_BASE_RELOCATION
00+4 VirtualAddress
04+4 SizeOfBlock
+2 TypeOffset
Type:4 Offset:12
relocation block
offset
PUSH EBP
PUSH offset szMyString
IMAGE_REL_BASED_HIGHLOW 3

driver

subsystem, checksum
low alignments mapping
different imports

normal.exe

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
000002FD	N/A	00000264	00000268	0000026C	00000270	00000274
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
kernel32.dll	1	00001090	00000000	00000000	000010F0	000010D0
msvcrt.dll	1	00001098	00000000	00000000	000010FD	000010D8

OFTs	FTs (IAT)	Hint	Name
00000298	000002D8	000002BE	000002C0
Dword	Dword	Word	szAnsi
000010BE	000010BE	0000	printf

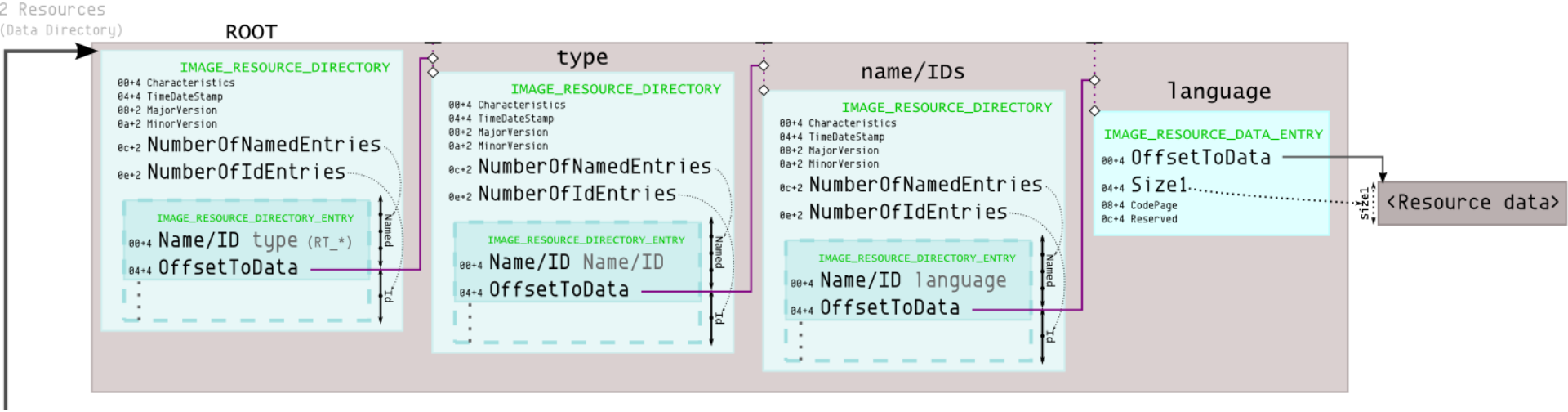
driver.sys

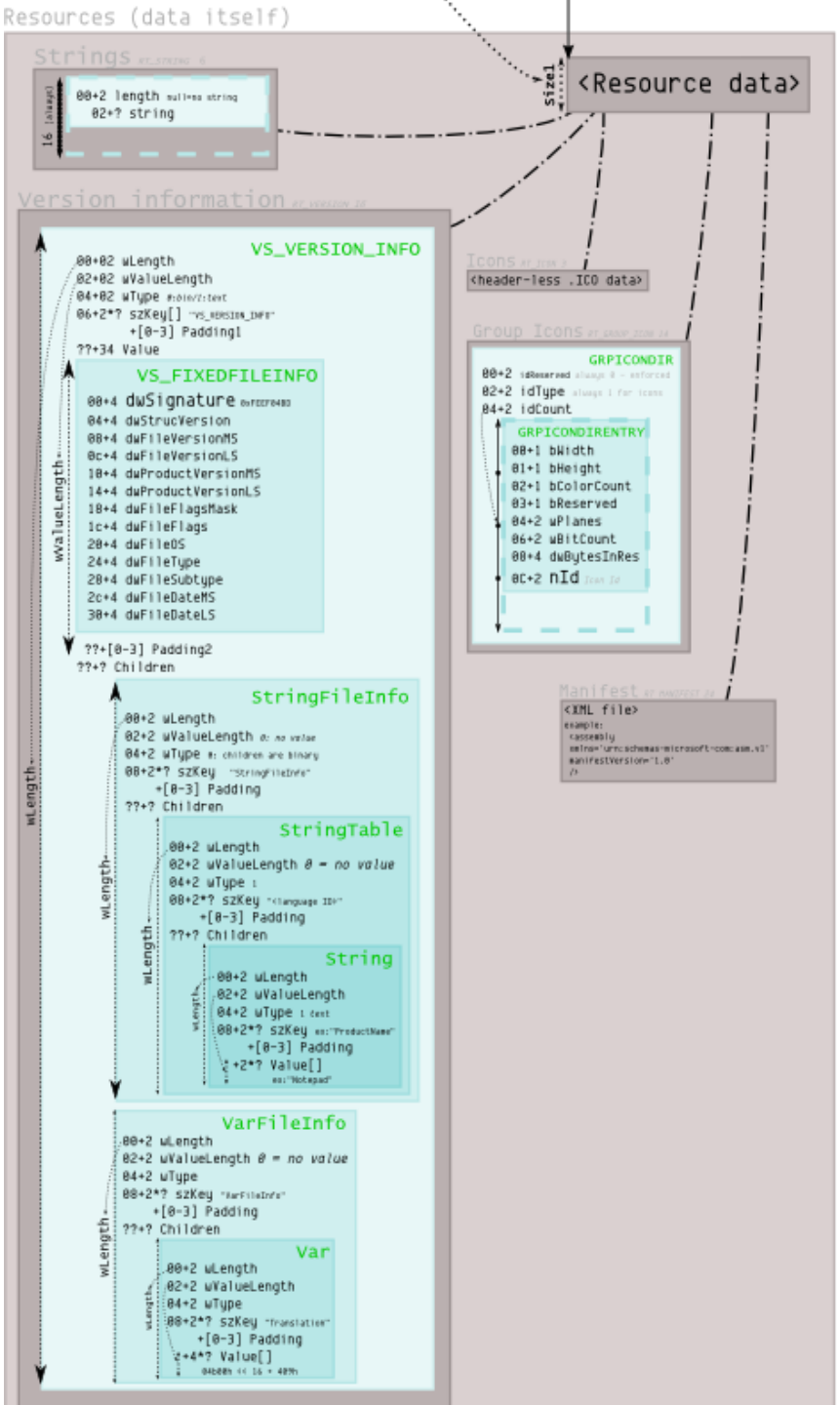
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
0000025B	N/A	00000220	00000224	00000228	0000022C	00000230
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
ntoskrnl.exe	1	00000248	00000000	00000000	0000025B	00000218

OFTs	FTs (IAT)	Hint	Name
00000248	00000218	00000250	00000252
Dword	Dword	Word	szAnsi
00000250	00000250	0000	DbgPrint

resources

structure
version, manifest/icon, APIs

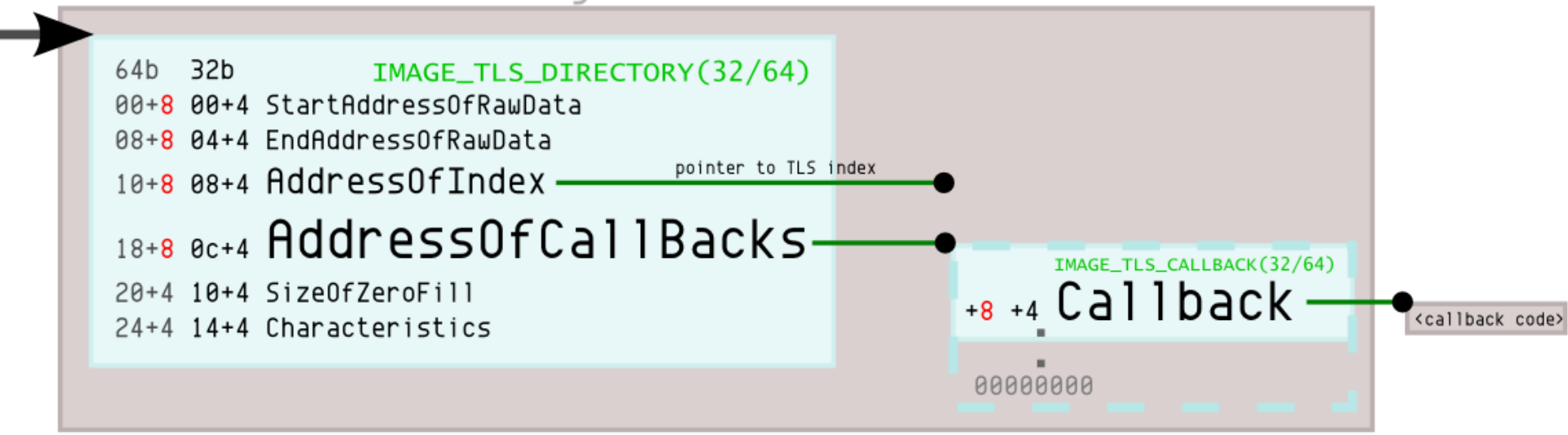

2 Resources
(Data Directory)
ROOT
IMAGE_RESOURCE_DIRECTORY
00+4 Characteristics
04+4 TimeDateStamp
08+2 MajorVersion
0a+2 MinorVersion
0c+2 NumberOfNamedEntries
0e+2 NumberOfIdEntries
IMAGE_RESOURCE_DIRECTORY_ENTRY
00+4 Name/ID type (RT_*)
04+4 OffsetToData
Named
Id
type
IMAGE_RESOURCE_DIRECTORY
00+4 Characteristics
04+4 TimeDateStamp
08+2 MajorVersion
0a+2 MinorVersion
0c+2 NumberOfNamedEntries
0e+2 NumberOfIdEntries
IMAGE_RESOURCE_DIRECTORY_ENTRY
00+4 Name/ID Name/ID
04+4 OffsetToData
Named
Id
name/IDs
IMAGE_RESOURCE_DIRECTORY
00+4 Characteristics
04+4 TimeDateStamp
08+2 MajorVersion
0a+2 MinorVersion
0c+2 NumberOfNamedEntries
0e+2 NumberOfIdEntries
IMAGE_RESOURCE_DIRECTORY_ENTRY
00+4 Name/ID language
04+4 OffsetToData
Named
Id
language
IMAGE_RESOURCE_DATA_ENTRY
00+4 OffsetToData
04+4 Size1
08+4 CodePage
0c+4 Reserved
Size1
<Resource data>


Resources (data itself)
Strings
00+2 length null=no string
02+? string
16 (always)
size1
<Resource data>
Version information
VS_VERSION_INFO
00+02 wLength
02+02 wValueLength
04+02 wType 0:bin/1:text
06+2*? szKey[] "VS_VERSION_INFO"
+[0-3] Padding1
??+34 Value
VS_FIXEDFILEINFO
00+4 dwSignature 0xFEEF04BD
04+4 dwStrucVersion
08+4 dwFileVersionMS
0c+4 dwFileVersionLS
10+4 dwProductVersionMS
14+4 dwProductVersionLS
18+4 dwFileFlagsMask
1c+4 dwFileFlags
20+4 dwFileOS
24+4 dwFileType
28+4 dwFileSubtype
2c+4 dwFileDateMS
30+4 dwFileDateLS
wValueLength
??+[0-3] Padding2
??+? Children
StringFileInfo
00+2 wLength
02+2 wValueLength 0: no value
04+2 wType 0: children are binary
08+2*? szKey "StringFileInfo"
+[0-3] Padding
??+? Children
StringTable
00+2 wLength
02+2 wValueLength 0 = no value
04+2 wType 1
08+2*? szKey "<language ID>"
+[0-3] Padding
??+? Children
String
00+2 wLength
02+2 wValueLength
04+2 wType 1 text
08+2*? szKey ex:"ProductName"
+[0-3] Padding
??+2*? Value[]
ex:"Notepad"
VarFileInfo
00+2 wLength
02+2 wValueLength 0 = no value
04+2 wType
08+2*? szKey "VarFileInfo"
+[0-3] Padding
??+? Children
Var
00+2 wLength
02+2 wValueLength
04+2 wType
08+2*? szKey "Translation"
+[0-3] Padding
??+4*? Value[]
04b0h << 16 + 4E4h
wLength
Icons
<header-less .ICO data>
Group Icons
GRPICONDIR
00+2 idReserved always 0 - enforced
02+2 idType always 1 for icons
04+2 idCount
GRPICONDIRENTRY
00+1 bWidth
01+1 bHeight
02+1 bColorCount
03+1 bReserved
04+2 wPlanes
06+2 wBitCount
08+4 dwBytesInRes
0C+2 nId icon id
Manifest
<XML file>
example:
<assembly
xmlns="urn:schemas-microsoft-com:asm.v1"
manifestVersion="1.0"
/>

Thread Local Storage

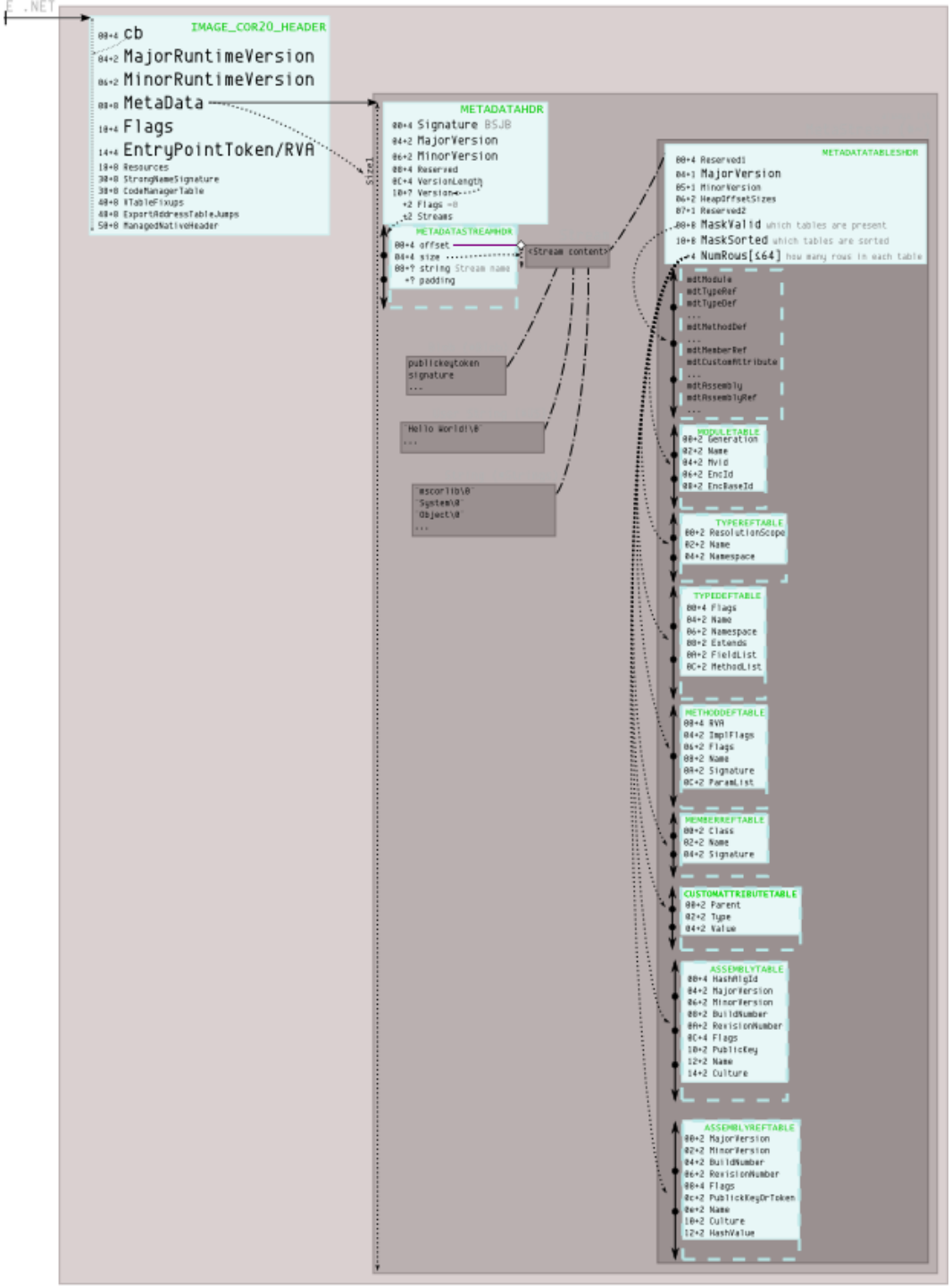
callback list
before EntryPoint & after ExitProcess

9 Thread Local Storage



.Net

different and integrated binary
2nd loader


E .NET
IMAGE_COR20_HEADER
cb
MajorRuntimeVersion
MinorRuntimeVersion
MetaData
Flags
EntryPointToken/RVA
Resources
StrongNameSignature
CodeManagerTable
VTableFixups
ExportAddressTableJumps
ManagedNativeHeader
METADATAHDR
Signature BSJB
MajorVersion
MinorVersion
Reserved
VersionLength
Version
Flags =0
Streams
METADATASTREAMHDR
offset
size
string Stream name
padding
<Stream content>
publickeytoken
signature
Hello World!\0
mscorlib\0
System\0
Object\0
METADATATABLESHDR
Reserved1
MajorVersion
MinorVersion
HeapOffsetSizes
Reserved2
MaskValid which tables are present
MaskSorted which tables are sorted
NumRows[≤64] how many rows in each table
mdtModule
mdtTypeRef
mdtTypeDef
mdtMethodDef
mdtMemberRef
mdtCustomAttribute
mdtAssembly
mdtAssemblyRef
MODULETABLE
Generation
Name
Mvid
EncId
EncBaseId
TYPEREFTABLE
ResolutionScope
Name
Namespace
TYPEDEFTABLE
Flags
Name
Namespace
Extends
FieldList
MethodList
METHODDEFTABLE
RVA
ImplFlags
Flags
Name
Signature
ParamList
MEMBERREFTABLE
Class
Name
Signature
CUSTOMATTRIBUTETABLE
Parent
Type
Value
ASSEMBLYTABLE
HashAlgId
MajorVersion
MinorVersion
BuildNumber
RevisionNumber
Flags
PublicKey
Name
Culture
ASSEMBLYREFTABLE
MajorVersion
MinorVersion
BuildNumber
RevisionNumber
Flags
PublicKeyOrToken
Name
Culture
HashValue

what about 64b?

very few changes

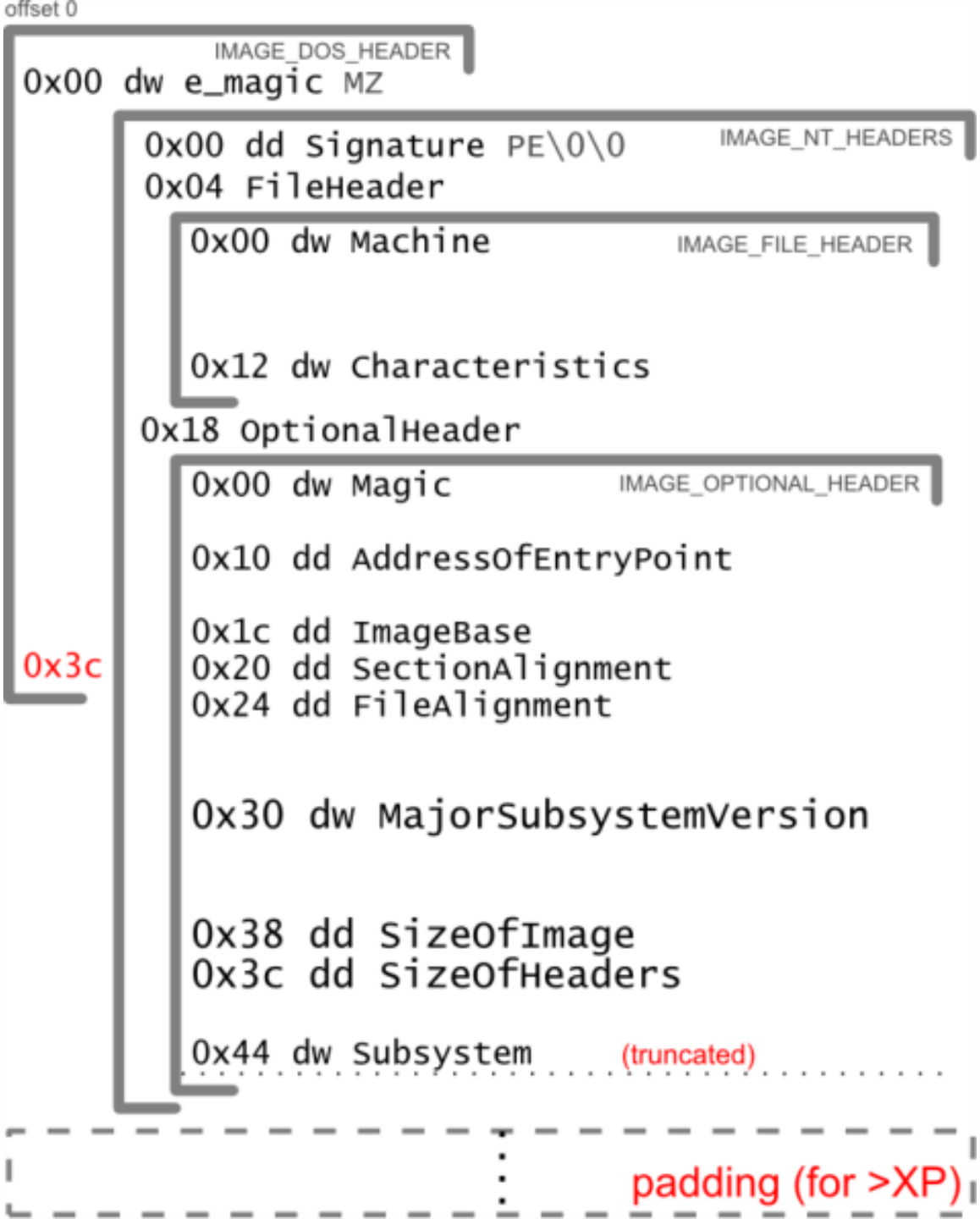
- 2 magic constants
- a few elements become QWord
  - ImageBase, Imports thunks, callbacks
- Exceptions have their own DataDirectory
  - no need for LoadConfig (~~SafeSEH~~)

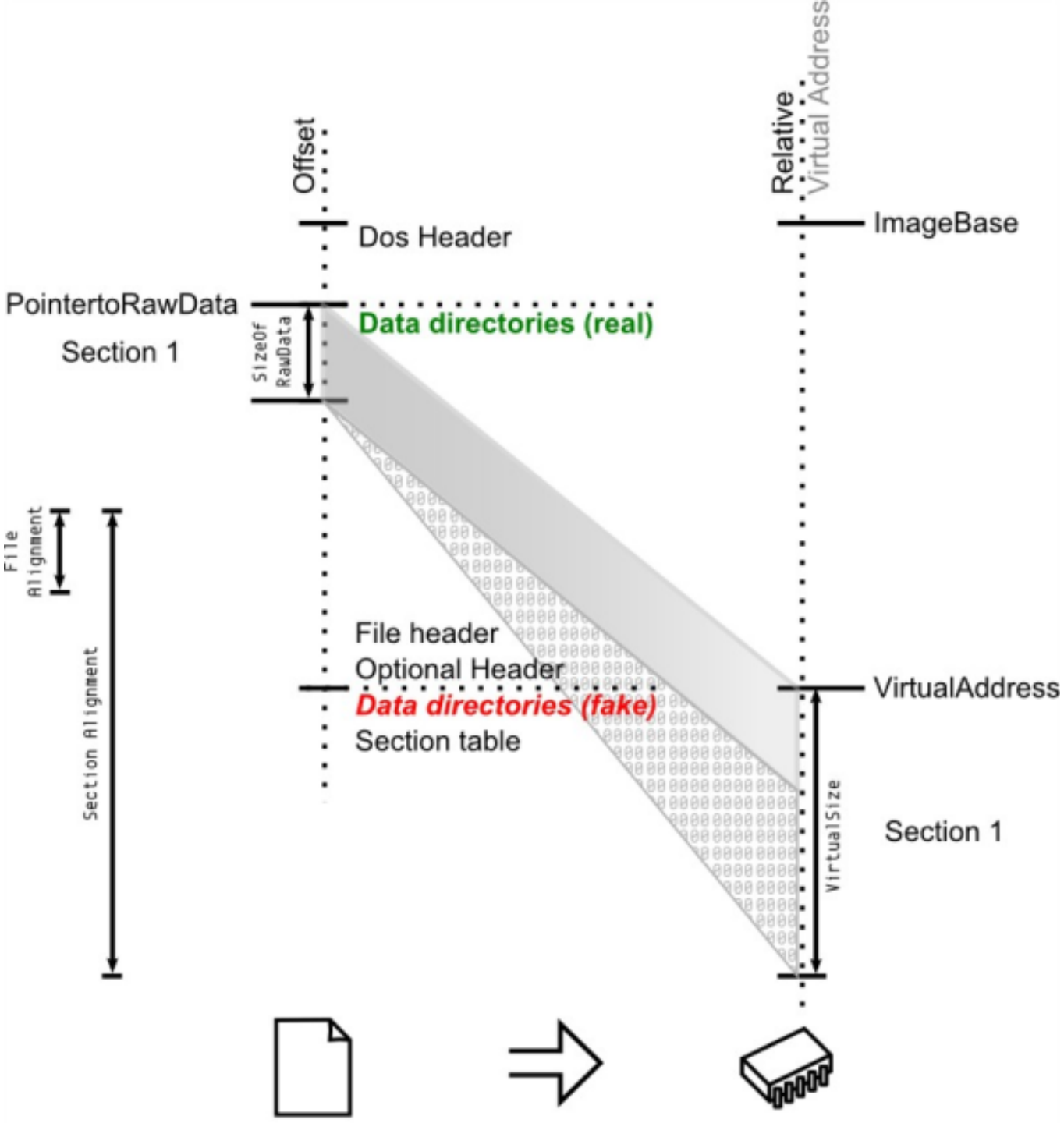
and ARM

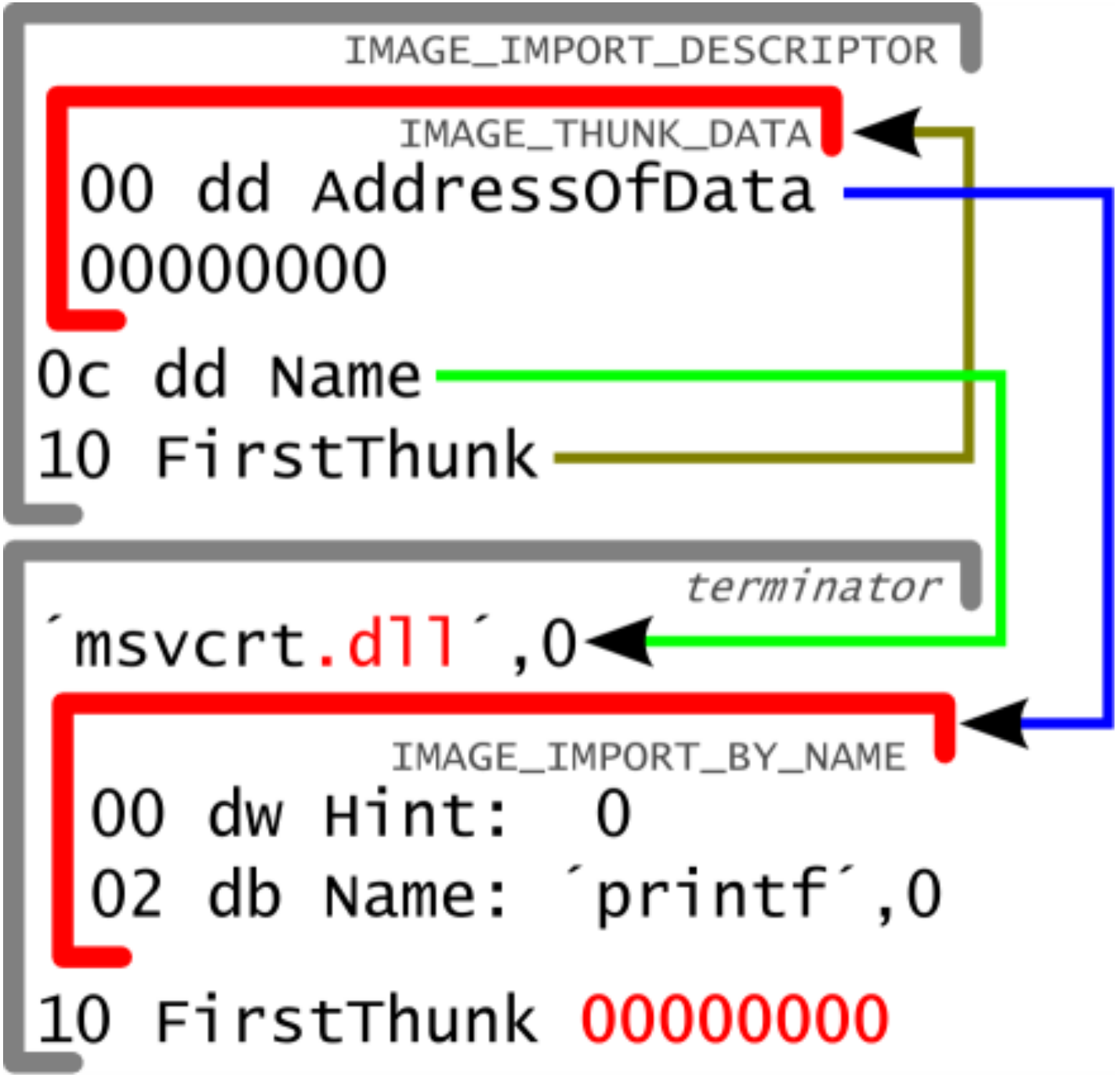
- a different magic constant
- still 16b DOS Stub !
- nothing special, PE wise
  - the beauty of ‘Portability’

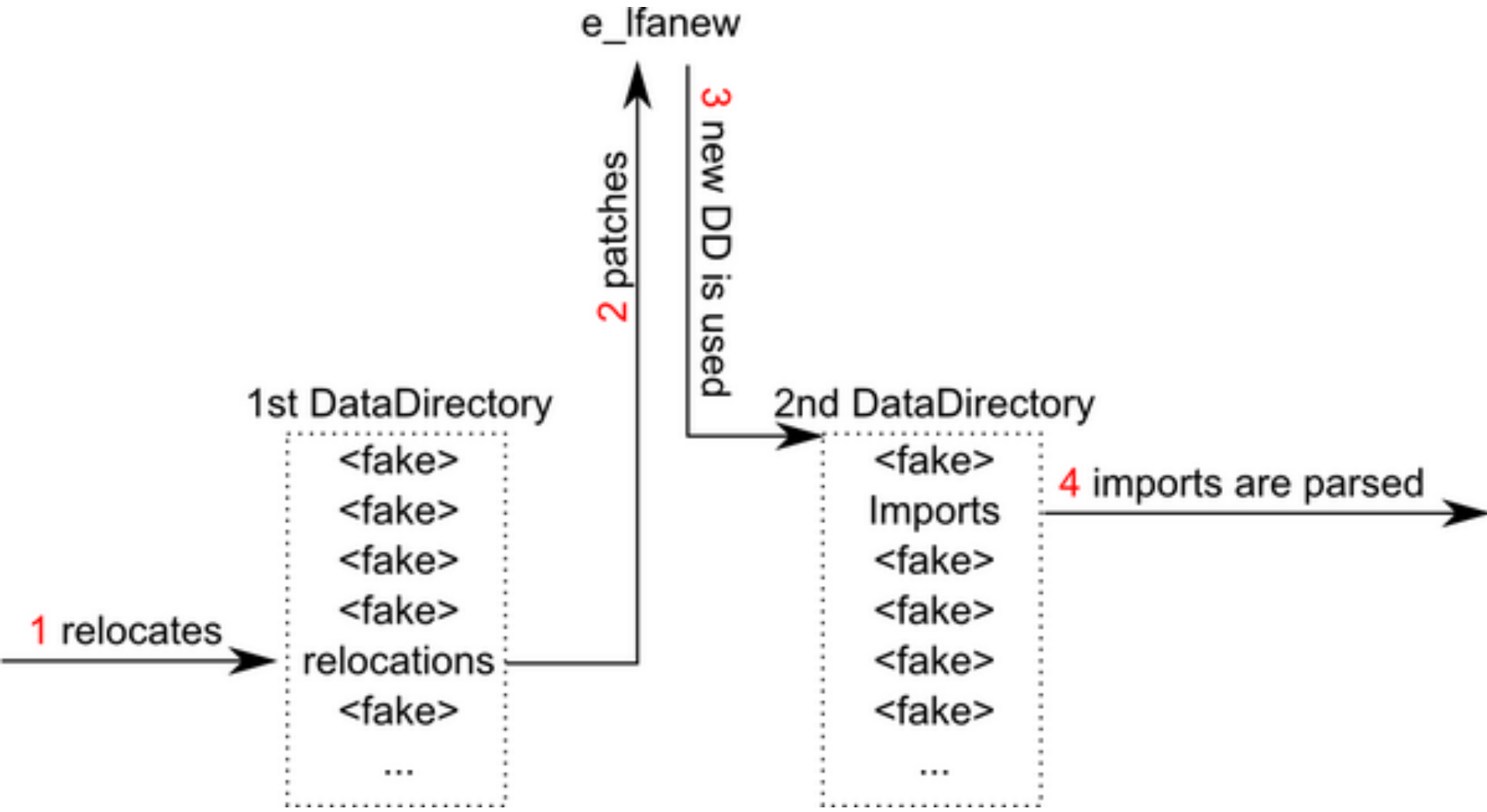
Classics

~~trivial~~


offset 0
IMAGE_DOS_HEADER
0x00 dw e_magic MZ
0x3c
IMAGE_NT_HEADERS
0x00 dd Signature PE\0\0
0x04 FileHeader
IMAGE_FILE_HEADER
0x00 dw Machine
0x12 dw Characteristics
0x18 OptionalHeader
IMAGE_OPTIONAL_HEADER
0x00 dw Magic
0x10 dd AddressOfEntryPoint
0x1c dd ImageBase
0x20 dd SectionAlignment
0x24 dd FileAlignment
0x30 dw MajorSubsystemVersion
0x38 dd SizeOfImage
0x3c dd SizeOfHeaders
0x44 dw Subsystem
(truncated)
padding (for >XP)


Offset
Relative Virtual Address
Dos Header
ImageBase
PointertoRawData
Section 1
SizeOf RawData
Data directories (real)
File Alignment
Section Alignment
File header
Optional Header
Data directories (fake)
Section table
VirtualAddress
VirtualSize
Section 1


IMAGE_IMPORT_DESCRIPTOR
IMAGE_THUNK_DATA
00 dd AddressOfData
00000000
0c dd Name
10 FirstThunk
terminator
´msvcrt.dll´,0
IMAGE_IMPORT_BY_NAME
00 dw Hint: 0
02 db Name: ´printf´,0
10 FirstThunk 00000000


e_lfanew
2 patches
3 new DD is used
1st DataDirectory
<fake>
<fake>
<fake>
<fake>
relocations
<fake>
...
1 relocates
2nd DataDirectory
<fake>
Imports
<fake>
<fake>
<fake>
<fake>
...
4 imports are parsed