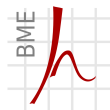


PE/COFF – IDA bevezető

Kódvisszafejtés.



Híradástechnikai Tanszék

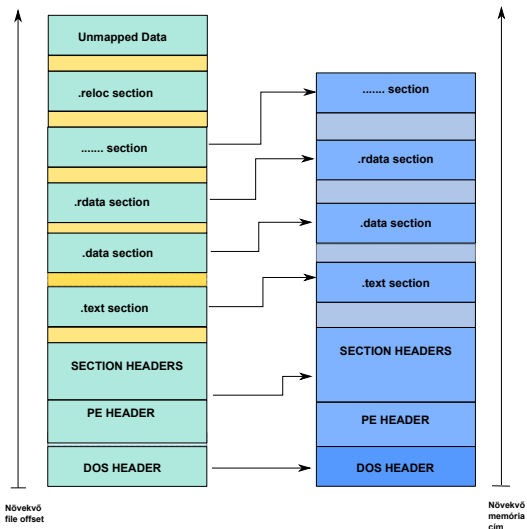
Izsó Tamás

2012. december 4.

Section 1

PE/COFF

Exe file leképzése a memóriába



Portable Executable fájl

- DOS operációs rendszerrel kompatibilis, figyelmeztető üzenetet ad, ha nem futtatható DOS alatt a program. A PE fájl eleje megegyezik a MZ végrehajtható fájlformátummal.
- DLL hívást támogatja. Az exe és a DLL azonos szerkezetű. A DLL fájlokat sokszor más kiterjesztéssel használják, pl. OCX , CPL
- Alpha, MIPS és .NET MSIL végrehajtható fájlformátuma is egyben.
- 64 bites programok hasonló formában vannak tárolva, egyes helyeken a 32 bitet 64 bites adatok váltották fel. PE32+ elnevezést kapta.

Relative Virtual Address (RVA)

A futtatható fájlt a loader nem másolja be a memóriába, hanem az egyes szekciókat memory mapped file-ként kezeli. A fájlban tárolt szekciók 512-vel (0x200) osztható byte határon kezdődnek, míg a memóriában új lapra kerülnek, ahol egy lap 4Kbyte (vagy 8 Kbyte). Ezért a fájl adott tartalma a fájl elejétől más távolságra van, mint ugyanez az adat a memóriába a program elejétől számítva.

RVA kiszámítása

A memóriában lévő címek a betöltés helyétől függetlenül vannak megadva. Ha egy cím a 0x4010DA címen szerepel és a program képe a 0x400000 címtől kezdve lett betöltve, akkor a relatív virtuális cím (RVA):

$$\text{RVA} = 0x4010DA - 0x400000 = 0x10DA$$

DOS HEADER

```

typedef struct _IMAGE_DOS_HEADER {           // DOS .EXE header
    WORD    e_magic;                          // Magic number
    WORD    e_cblp;                           // Bytes on last page of file
    WORD    e_cp;                              // Pages in file
    WORD    e_crlc;                           // Relocations
    WORD    e_cparhdr;                        // Size of header in paragraphs
    WORD    e_minalloc;                       // Minimum extra paragraphs needed
    WORD    e_maxalloc;                       // Maximum extra paragraphs needed
    WORD    e_ss;                             // Initial (relative) SS value
    WORD    e_sp;                             // Initial SP value
    WORD    e_csum;                           // Checksum
    WORD    e_ip;                             // Initial IP value
    WORD    e_cs;                             // Initial (relative) CS value
    WORD    e_lfarlc;                         // File address of relocation table
    WORD    e_ovno;                           // Overlay number
    WORD    e_res[4];                         // Reserved words
    WORD    e_oemid;                          // OEM identifier (for e_oeminfo)
    WORD    e_oeminfo;                        // OEM information; e_oemid specific
    WORD    e_res2[10];                       // Reserved words
    LONG    e_lfanew;                         // File address of new exe header
} IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;

```

DOS HEADER tartalma

- Két érdemi részt tartalmaz:
 - 1 e_magic értéke "MZ" (Mark Zbikowski)
 - 2 e_lfanew file offset, a PE header-re mutat.
- DOS_HEADER után kis DOS program következnek.
- A PE rész közvetlenül ez után, 8-cal osztható címre igazítva található.

DOS header dumpja

00000000:	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ	DOS HEADER
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 C0 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:	C9 B5 76 DC 8D D4 18 8F 8D D4 18 8F 8D D4 18 8F	. . v	
00000090:	84 AC 8B 8F 8E D4 18 8F 8D D4 19 8F 8F D4 18 8F	
000000A0:	84 AC 9B 8F 8C D4 18 8F 84 AC 89 8F 8C D4 18 8F	
000000B0:	52 69 63 68 8D D4 18 8F 00 00 00 00 00 00 00 00	Rich	NT HEADER
000000C0:	50 45 00 00 4C 01 02 00 9C 0B A8 50 00 00 00 00	PE . L P	
000000D0:	00 00 00 00 E0 00 03 01 0B 01 09 00 00 02 00 00	
000000E0:	00 02 00 00 00 00 00 00 00 10 00 00 00 10 00 00	

NT HEADER

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

DOS header dumpja

```

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 C0 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 73 6F 67 73 61 6D 20 63 61 6F 6F 6F
00000060: 74 20 ( Intel 386 or later processors and compatible processors )
00000070: 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00
00000080: C9 signature C 8D D4 18 8F 8D D4 18 8F 8D D4 18 8F
00000090: 84 AC 8B 8F 8E D4 18 8F 8D D4 18 8F 8D D4 18 8F
000000A0: 84 AC 9B 8F 8C D4 18 8F 8D D4 18 8F 8D D4 18 8F
000000B0: 52 69 63 68 8D D4 18 8F 00 00 00 00 00 00 00 00
000000C0: 50 45 00 00 4C 01 02 00 9C 0B A8 50 00 00 00 00
000000D0: 00 00 00 00 E0 00 03 01 0B 01 09 00 00 02 00 00
000000E0: 00 02 00 00 00 00 00 00 00 10 00 00 00 10 00 00

```

DOS HEADER

NT HEADER

szimbólumok
száma

opcionális header
mérete

IMAGE_FILE_EXECUTABLE_IMAGE
IMAGE_FILE_RELOCS_STRIPPED

szimbólumtáblára
mutató pointer

IMAGE FILE HEADER

```
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 ;
```

IMAGE OPTIONAL HEADER – standard mezők

```
typedef struct _IMAGE_OPTIONAL_HEADER {  
    WORD        Magic ;  
    BYTE        MajorLinkerVersion ;  
    BYTE        MinorLinkerVersion ;  
    DWORD       SizeOfCode ;  
    DWORD       SizeOfInitializedData ;  
    DWORD       SizeOfUninitializedData ;  
    DWORD       AddressOfEntryPoint ;  
    DWORD       BaseOfCode ;  
    DWORD       BaseOfData ;
```

IMAGE_OPTIONAL_HEADER – NT specifikus mezők

```
DWORD ImageBase ;
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 ;
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 ;
```

Optional header dumpja

Address	Bytes	Annotations	Structure
000000C0:	50 45 00 00 4C 01 02 00 9C 0B A8 50 00 00 00 00	.data mérete, .bbs mérete, program belépési pontja RVA, PE32, Kód címe RVA	PE...L.....P....
000000D0:	00 00 00 00 E0 00 03 01 0B 01 09 00 00 02 00 00	
000000E0:	00 02 00 00 00 00 00 00 00 10 00 00 00 10 00 00	
000000F0:	00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00	
00000100:	05 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00	
00000110:	00 30 00 00 00 04 00 00 00 00 00 00 00 03 00 84	
00000120:	adat címe RVA, image betöltési címe, címhatárra igazítás, file offset határra igazítás	
00000130:		
00000140:	0C 20 00 00 28 00 00 00 00 00 00 00 00 00 00 00	
00000150:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000160:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000170:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000180:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000190:	00 00 00 00 00 00 00 00 00 20 00 00 0C 00 00 00	
000001A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000001B0:	00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00	text...

SECTION TABLE

A section táblák kezdete:

```
pSectionTable = &NtHeader.OptionalHeader +
                NtHeader.FileHeader.SizeOfOptionalHeader;
```

```
typedef struct _IMAGE_SECTION_HEADER {
    BYTE    Name[IMAGE_SIZEOF_SHORT_NAME]; // section neve
    union {
        DWORD    PhysicalAddress;
        DWORD    VirtualSize;           // lefoglalt méret
    } Misc;
    DWORD    VirtualAddress;           // RVA
    DWORD    SizeOfRawData;           // adatok tényleges mérete
    DWORD    PointerToRawData;
    DWORD    PointerToRelocations;
    DWORD    PointerToLinenumbers;
    WORD     NumberOfRelocations;
    WORD     NumberOfLinenumbers;
    DWORD    Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```


SECTION TABLE dumpja

opcionális header mérete + opcionális header kezdete

section tábla kezdete

NT HEADER

IMAGE DATA DIRECTORY

SECTION TABLE

data

adatok méret

helye a fájlban

cím (RVA)

data

adatok

.text adatok

```

000000c0: 50 45 00 00 4c 01 02 00 9c 86 a8 50 00 00 00 00 PE...L...P
000000d0: 00 00 00 00 0e 00 03 01 08 01 09 00 00 02 00 00
000000e0: 00 02 00 00 00 00 00 00 00 10 00 00 00 02 00 00
000000f0: 00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00
00000100: 05 00 00 00 00 00 00 05 00 00 00 00 00 00 00 00
00000110: 00 30 00 00 04 00 00 00 00 00 03 00 00 84 00 00
00000120: 00 00 10 00 00 10 00 00 00 10 00 00 00 10 00 00
00000130: 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000140: 0c 20 00 00 28 00 00 00 00 00 00 00 00 00 00 00
00000150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000190: 00 00 00 00 00 00 00 00 20 00 00 0c 00 00 00 00
000001a0: 00 00 00 00 00 00 00 00 0e 74 83 78 04 00 00 00
000001b0: 00 00 00 00 00 00 00 00 2e 74 83 78 04 00 00 00
000001c0: 2a 00 00 00 10 00 00 00 00 02 00 00 00 04 00 00
000001d0: 00 00 00 00 00 00 00 00 00 00 00 20 00 20 00 60
000001e0: 2e 72 64 61 74 61 00 00 00 5c 00 00 00 20 00 00
000001f0: 00 02 00 00 00 06 00 00 00 00 00 00 00 00 00 00
00000200: 00 00 00 00 40 00 00 40 00 00 00 00 00 00 00 00
00000210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000250: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000260: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000300: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000310: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000320: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000330: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000340: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000350: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000360: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000370: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000380: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000390: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000400: 55 88 ec 51 6a 04 6a 03 ff 15 20 40 00 83 c4 u..Qj.
00000410: 08 89 45 fc 88 45 fc 50 ff 15 04 20 40 00 83 c4
00000420: 04 88 01 00 00 00 00 88 e5 5d c3 00 00 00 00 00
00000430: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

IMAGE DATA DIRECTORY – 16 db bejegyzés

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

IMAGE DATA Bejegyzések – 16 db bejegyzés

0	Export Table	dll .edata Section
1	Import Table	dll .idata
2	Resource Table	.rsrc Section
3	Exception Table	.pdata Section
4	Certificate Table	
5	Base Relocation Table	.reloc Section
6	Debug	.debug Section
7	Architecture Specific Data	reserved 0
8	Global Ptr	
9	TLS Directory	
10	Load Configuration Directory	
11	Bound Import	
12	IAT	import address table
13	Delay Load Import Descriptors	
14	CLR Runtime descriptor	
15	reserved	0

IMAGE DATA kikeresése a SECTION TABLE-ban

A section tábla kezdete a memóriában (RVA):

```
PIMAGE_SECTION_HEADER getSectionHdr( DWORD rva )
{
    PIMAGE_SECTION_HEADER section = IMAGE_FIRST_SECTION(pNTHHeader);
    unsigned i;

    for(i=0; i < pNTHHeader->FileHeader.NumberOfSections; i++, section++) {
        DWORD size = section->Misc.VirtualSize;

        // Is the RVA within this section?
        if( (rva >= section->VirtualAddress) &&
            (rva < (section->VirtualAddress + size)))
            return section;
    }
    return 0;
}
```

A section tábla kezdete a fájlban (offset):

```
DWORD GetOffsetFromRVA( DWORD rva , PIMAGE_SECTION_HEADER pSectionHdr)
{
    // rva = 0x200C
    // Dumpban 0x2000 - 0x600, delta = 0x1a00
    DWORD delta = pSectionHdr->VirtualAddress - pSectionHdr->PointerToRawData;
    return rva - delta ; // 0x60C
}
```

EXPORT DIRECTORY Entry

```

typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD    Characteristics;           // Set 0
    DWORD    TimeDateStamp;
    WORD     MajorVersion;             // User set
    WORD     MinorVersion;            // User set
    DWORD    Name;                     // DLL name (RVA)
    DWORD    Base;
    DWORD    NumberOfFunctions;
    DWORD    NumberOfNames;
    DWORD    AddressOfFunctions;       // EAT (RVA)
    DWORD    AddressOfNames;          // RVA
    DWORD    AddressOfNameOrdinals;   // RVA
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;

```

IMPORT DIRECTORY DATA dumpja

Address	Hex Data	Dec Data	Comment
000000c0:	50 45 00 00 4c 01 02 00 9c 08 a8 50 00 00 00 00	PE...L...P...	
000000d0:	00 00 00 00 e0 00 03 01 08 01 09 00 02 00 00 00		
000000e0:	00 02 00 00 00 00 00 00 10 00 00 00 10 00 00		
000000f0:	00 20 00 00 00 00 40 00 00 10 00 00 02 00 00		
00000100:	05 00 00 00 00 00 00 00 05 00 00 00 00 00 00		
00000110:	00 30 00 00 00 04 00 00 00 00 00 03 00 84 00		
00000120:	00 00 10 00 00 10 00 00 00 00 10 00 00 00 00		
00000130:	00 00 00 00 10 00 00 00 00 00 00 00 00 00 00		
00000140:	0c 20 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000150:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000160:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000170:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000180:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000190:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000001a0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000001b0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000001c0:	2a 00 00 00 00 00 00 00 02 00 00 00 04 00 00	text	
000001d0:	00 00 00 00 00 00 00 00 00 00 00 20 00 60 00	data	
000001e0:	2e 72 64 61 74 61 00 00 0c 00 00 00 20 00 00		
000001f0:	00 02 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000200:	00 00 00 00 40 00 00 00 00 00 00 00 00 00 00		
00000210:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000220:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000230:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000240:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000250:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000260:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000270:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000280:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000290:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000002a0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000002b0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000002c0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000002d0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000002e0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000002f0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000300:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000310:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000320:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000330:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000340:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000500:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000510:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000520:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000530:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000540:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000550:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000560:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000570:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000580:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000590:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000005a0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000005b0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000005c0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000005d0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000005e0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000005f0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000600:	4c 20 00 00 40 20 00 00 00 00 00 34 20 00 00	L.....R	
00000610:	00 00 00 00 00 00 00 00 52 20 00 00 20 00 00		
00000620:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000630:	00 00 00 00 4c 20 00 00 40 20 00 00 00 00 00		
00000640:	01 00 46 75 6e 63 74 69 6f 6e 00 00 41 64 00	Function	Ad
00000650:	64 00 63 61 6c 63 2e 64 6c 6c 00 00 00 00 00	d.calc.dll	
00000660:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	d.calc.dll	
00000670:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000680:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000690:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000006a0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000006b0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000006c0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000006d0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000006e0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
000006f0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000700:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000710:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000720:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000730:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000740:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000750:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000760:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000770:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
00000780:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		

IMAGE_IMPORT_DESCRIPTOR

```

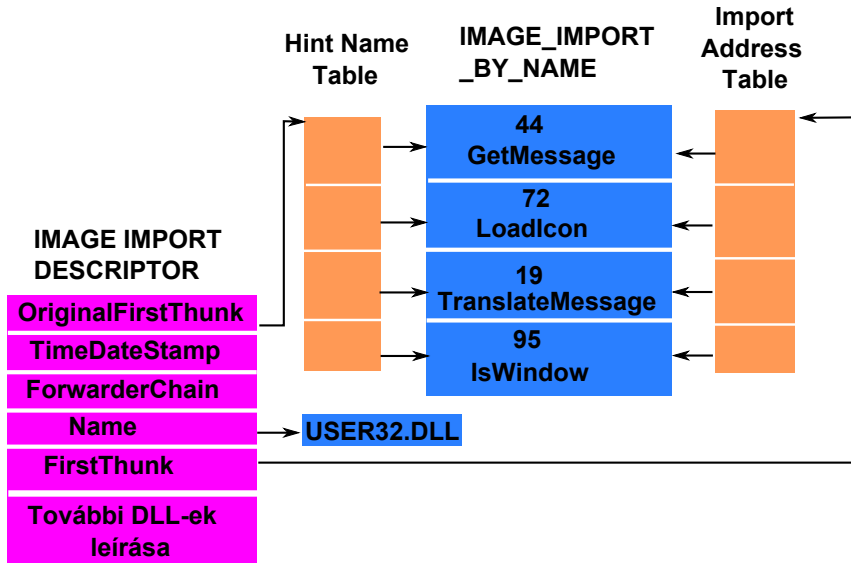
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        DWORD    Characteristics;
        DWORD    OriginalFirstThunk; // RVA to original unbound IAT
    } DUMMYUNIONNAME;
    DWORD    TimeDateStamp;           // 0 if not bound,

    DWORD    ForwarderChain;         // -1 if no forwarders
    DWORD    Name;
    DWORD    FirstThunk;             // RVA to IAT
} IMAGE_IMPORT_DESCRIPTOR;

typedef struct _IMAGE_IMPORT_BY_NAME {
    WORD    Hint;
    BYTE    Name[1];
} IMAGE_IMPORT_BY_NAME, *PIMAGE_IMPORT_BY_NAME;

```

IMPORT TABLE



IMPORT TABLE

00000560:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000570:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000580:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000590:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000005A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000005B0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000005C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000005D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000005E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000005F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000600:	4C 20 00 00 40 20 00 00 00 00 00 00 34 20 00 00	L.....
00000610:	00 00 00 00 00 00 00 00 52 20 00 00 00 20 00 00R.....
00000620:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000630:	00 00 00 00 4C 20 00 00 40 20 00 00 00 00 00 00L.....
00000640:	01 00 46 75 6E 63 74 69 6F 6E 00 00 00 00 41 64	..Function...Ad
00000650:	64 00 63 61 6C 63 2E 64 6E 15C 00 00 00 00 00 00	d.calc.dll.....
00000660:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Ordinal 0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000680:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000690:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006B0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000700:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

.rdata
adatok

OriginalFirstThunk

2

Ordinal

Name (RVA)

Name (RVA)

FirstThunk

Importált
szimbólum
(függvény)
neve

IMPORT TABLE – program indítása után a memóriában

```

00402000  00 10 00 10|30 10 00 10|00 00 00 00|34 20 00 00| ....0.....4 ..
00402010  00 00 00 00|00 00 00 00|52 20 00 00|00 20 00 00| .....R ... ..
00402020  00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00| .....
00402030  00 00 00 00|4C 20 00 00|40 20 00 00|00 00 00 00| ....L ... .....
00402040  01 00 46 75|6E 63 74 69|6F 6E 00 00|00 00 41 64| ..Function....Ad
00402050  64 00 63 61|6C 63 2E 64|6C 6C 00 00|00 00 00 00| d.calc.dll.....

```

DLL-ben lévő függvény hívása

```

00401000    55                PUSH EBP
00401001    8BEC             MOV EBP,ESP
00401003    51                PUSH ECX
00401004    6A 04            PUSH 4
00401006    6A 03            PUSH 3
00401008    FF15 00204000    CALL DWORD PTR DS:[<&calc.Add >]
0040100E    83C4 08          ADD ESP,8
00401011    8945 FC          MOV DWORD PTR SS:[LOCAL.1],EAX
00401014    8B45 FC          MOV EAX,DWORD PTR SS:[LOCAL.1]
00401017    50                PUSH EAX
00401018    FF15 04204000    CALL DWORD PTR DS:[<&calc.Function >]
0040101E    83C4 04          ADD ESP,4
00401021    B8 01000000      MOV EAX,1
00401026    8BE5             MOV ESP,EBP
00401028    5D                POP EBP
00401029    C3                RETN

```

DLL tartalma

```

10001000 55          PUSH EBP
10001001 8BEC        MOV EBP,ESP
10001003 8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
10001006 0345 0C     ADD EAX,DWORD PTR SS:[EBP+0C]
10001009 5D          POP EBP
1000100A C3          RETN

10001010 55          PUSH EBP
10001011 8BEC        MOV EBP,ESP
10001013 8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
10001016 2B45 0C     SUB EAX,DWORD PTR SS:[EBP+0C]
10001019 5D          POP EBP
1000101A C3          RETN

10001020 55          PUSH EBP
10001021 8BEC        MOV EBP,ESP
10001023 8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
10001026 0FAF45 0C   IMUL EAX,DWORD PTR SS:[EBP+0C]
1000102A 5D          POP EBP
1000102B C3          RETN

10001030 55          PUSH EBP
10001031 8BEC        MOV EBP,ESP
10001033 8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
10001036 50          PUSH EAX
10001037 68 00C00010 PUSH OFFSET 1000C000 ; ASCII "Printf from DLL %d "
1000103C E8 7F000000 CALL 100010C0
10001041 83C4 08     ADD ESP,8
10001044 5D          POP EBP
10001045 C3          RETN

```

Importált függvények címének feloldása

- A program a dll-ben lévő **Add()** függvényt a 00401008 címen a **CALL [00402000]** utasítással hívja meg. Az indirekten hívott függvény címét az Import Address Table (IAT) tárolja.
- A fordító a `__declspec(dllimport) int Add(int,int)`; függvény *tárolási osztály módosító* hatására generál optimálisabb kódot. Ha nem adjuk meg, akkor a **CALL XXXXXXXX** utasításnak meg kellene hívni egy thunk kódot, ami megoldja a DLL-ben lévő függvény hívását. A fordító egy `__imp__Add` szimbólumot is generál, amely egy pointer, és az IAT táblában foglal helyet.
- A loadernek a betöltésnél csak az IAT-t kell megváltoztatni.
- A függvényeket nemcsak név, hanem sorszám (ordinal number) alapján is meg lehet hívni.

Optimalizálatlan DLL függvényhívás

Ha a főprogramban nem használjuk a `__declspec(dllimport)` függvény *tárolási osztály módosítót*, ekkor a következő nem annyira hatékony kód keletkezik:

```

00401000  55          PUSH EBP
00401001  8BEC       MOV EBP,ESP
00401003  51          PUSH ECX
00401004  6A 04      PUSH 4
00401006  6A 03      PUSH 3
00401008  E8 21000000 CALL <JMP.&calc.Add> ; Jump to calc.Add
0040100D  83C4 08    ADD ESP,8
00401010  8945 FC    MOV DWORD PTR SS:[LOCAL.1],EAX
00401013  8B45 FC    MOV EAX,DWORD PTR SS:[LOCAL.1]
00401016  50          PUSH EAX
00401017  E8 0C000000 CALL <JMP.&calc.Function> ; Jump to calc.Function
0040101C  83C4 04    ADD ESP,4
0040101F  B8 01000000 MOV EAX,1
00401024  8BE5       MOV ESP,EBP
00401026  5D          POP EBP
00401027  C3          RETN
00401028  $- FF25 04204000 JMP DWORD PTR DS:[&calc.Function >]
0040102E  $- FF25 00204000 JMP DWORD PTR DS:[&calc.Add >]

```

BIND utility

```

Bind.Exe -v -u test_implicit.exe
BIND: test_implicit.exe - Imports from calc.dll
BIND: test_implicit.exe - Add Bound to 0000000010001000
BIND: test_implicit.exe - Function Bound to 0000000010001030
BIND: Details of binding of test_implicit.exe
      Import from calc.dll [50a80afb]

```

```

00000600: 00 10 00 10 30 10 00 10 00 00 00 00 34 20 00 00 .....
00000610: FF FF FF FF FF FF FF FF 52 20 00 00 00 20 00 00 .....R.....
00000620: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000630: 00 00 00 00 4C 20 00 00 40 20 00 00 00 00 00 00 ....L.....
00000640: 01 00 46 75 6E 63 74 69 6F 6E 00 00 00 00 41 64 ..Function...Ad
00000650: 64 00 63 61 6C 63 2E 64 6C 6C 00 00 00 00 00 00 d.calc.dll.....

```

A Bind utility feloldja az IAT táblában levő címeket. Hogyan lehetséges ez?

- A program relokálása a virtuális memóriakezelés által hardware szintű támogatást kap.
- Az időbélyeg és az Image Bound Import Descriptor (lásd az irodalmat) alapján a loader megbízhatóan el tudja dönteni, hogy a DLL fájl újra lett linkelve.

Irodalom

- 1** Matt Pietrek An In-Depth Look into the Win32 Portable Executable File Format <http://msdn.microsoft.com/en-us/magazine/bb985992.aspx>
- 2** Matt Pietrek Peering Inside the PE: A Tour of the Win32 Portable Executable File Format <http://msdn.microsoft.com/en-us/magazine/ms809762.aspx>
- 3** Microsoft PE and COFF Specification. <http://msdn.microsoft.com/en-us/library/windows/hardware/gg463119.aspx>

Section 2

IDA bevezető

IDA decompiler

- Program szerzője Ilfak Guilfanov, ma már többen dolgoznak rajta.
- Honlap: <http://www.hex-rays.com/>
- IDA demo illetve freeware letölthető a <http://www.hex-rays.com/products/ida/support/download.shtml> helyről.

Objektumorientált kód visszafejtése

A 11-edik előadáson 8 és 12 diákon lévő program visszafejtése után 34952 sor keletkezett, amiből 18260 (vannak kommentek is) kód a többi adat. Első feladat a főprogram megkeresése.

Mit tudunk:

- A `int main(int argc, char*argv[], char* env[])` program 3 paramétert vesz át.
- 2. paraméter a parancssori argumentum. A Visual Studio-val fordított kódban a `mainCRTStartup` kód hívja meg a `main` függvényt, amit a `C:\Program Files\Microsoft Visual-Studio 9.0\VC\crt\src\crt0.c`-hez hasonló directoryban találunk meg, a verziószámtól függően.
- A parancssori argumentumokat a `GetCommandLine` Windows API függvényvel érjük el.

main program kikeresése

- IDA-ba válasszuk ki <Search> menü alatt <Text> menüpontot, és keressük meg a GetCommandLine függvényt. Mivel ez DLL-ben lévő exportált függvény, ezért a hívó Import Name Table adatában megtalálható a függvény szimbólikus neve, ugyanakkor a main szimbólikus név már kikerült a lefordított kódból.
- Keressünk a függvényhívás közelébe olyan függvényt, aminek három paramétere van.

```
.text:004013F6      mov     eax, dword_40DF78
.text:004013FB      mov     dword_40DF7C, eax
.text:00401400      push   eax
.text:00401401      push   dword_40DF70
.text:00401407      push   dword_40DF6C
.text:0040140D      call   sub_401000
```

Navigálás és átnevezés

- Egér gomb dupla klikk-kel a `sub_401000` függvény nevére kattintva a függvény definíciójához ugorhatunk. Ez *escape* gomb hatására visszatérhetünk az előző programrészhez.
- `sub_401000 proc near ; CODE XREF: start-5C` sorra állva nyomjuk meg az N billentyűt, vagy a jobb egérgomb által előhívott menüből válasszuk ki a <Rename> almenüt, és nevezzük át a függvényt. Térjünk vissza a hívó részhez. Ezt megtehetjük a CODE XREF keresztreferencia részben megadott hívási helyek (jelenleg csak egy) egyikére kattintva. Az adott címen már a **call** main utasítás szerepel. Az IDA a szimbólikus neveket továbbterjeszti, ameddig a program alapján követhető a rá való hivatkozás.

Függvény prototípusának a megadása

Térjünk vissza a main függvényhez, és a jobb egérgomb <Set function type> vagy az Y billentyűvel adjuk meg a függvény prototípusát.

```
int __cdecl main(int argc, char* argv [], char* env[] );
```

Eredmény:

```
.text:00401000 ; int __cdecl main(int argc,char **argv,char **env)
.text:00401000 main      proc near                ; CODE XREF: start-5C
.text:00401000 var_4    = dword ptr -4
.text:00401000 argc     = dword ptr 8
.text:00401000 argv    = dword ptr 0Ch
.text:00401000 env     = dword ptr 10h

.text:00401024          cmp          [ebp+argc], 2
```

Valamint a hívott helyen:

```
.text:00401400          push    eax                ; env
.text:00401401          push    argv                ; argv
.text:00401407          push    argc                ; argc
.text:0040140D          call    main
```

Struktúra (class) definiálás

- 1 Nyissuk meg a *Structures* dialogus ablakot a shift-F9 gombbal, vagy a <View> <Open subview> <Structures> menü alapján.
- 2 Hozzunk létre új struktúrát az *Ins* gombbal, és a *Create structure* ablakban nevezzük el az új struktúrát.
- 3 Legyen a neve *C_class*. Az OK gomb megnyomása után létrejön egy üres struktúra.

```
00000000 C_class          struc ; (sizeof=0x0)
00000000 C_class          ends
```

Struktúra tagváltozóinak a megadása

- 1 A struktúra végére állva a D billentyűvel vehetünk fel új tagváltozókat. Ismételt nyomkodásával az adat méretét változtathatjuk meg. Ha nem találjuk meg a megfelelő méretet, akkor az <Option> <Setup datatype> vagy a Alt-D billentyű lenyomásával hívhatunk elő egy ablakot, amiben be lehet állítani nagyobb méretű változók megjelenését is.
- 2 Az N gombbal átnevezhetjük a tagváltozókat.

```

00000000 C_class          struc ; (sizeof=0xA)
00000000 c_vtptr        dw ?
00000002 a              dw ?
00000004 b_thunk_vtptr dw ?
00000006 b              dw ?
00000008 c              dw ?
0000000A C_class       ends

```


Struktúra memóriaterülethez rendelése

- 1 A `var_24` stack frame tartalmazza a `C` c; objektumot. Az adott értékre duplán kattintva bejön a *Stack frame* ablak.
- 2 Az `<Edit>` `<Struct var>` menük, vagy az `Alt-Q` billentyű segítségével hívjuk be a *"choose a structure"* ablakot és válasszuk ki a megfelelő struktúra definíciót.

```
.text:00401000 main      proc near      ; CODE XREF: start-5C
.text:00401000
.text:00401000 var_28    = dword ptr -28h
.text:00401000 c        = C_class ptr -24h
.text:00401000 var_10   = dword ptr -10h
.text:00401000 var_C    = dword ptr -0Ch
.text:00401000 var_4    = C_class ptr -4
```

Következő óra

- 1 Bináris fájlformátum visszafejtése.
- 2 Dinamikus nyomkövetés Intel PIN tool-lal.
- 3 Egyszerű Ida plugin írás.