Formaty plików wykonywalnych dla systemu DOS, Windows i Linux

Mateusz Gorządek i Arkadiusz Grzanka

Formaty plików wykonywalnych dla systemu DOS, Windows i Linux

Mateusz Gorządek i Arkadiusz Grzanka

Plik wykonywalny

Plik, który może być uruchomiony bezpośrednio w środowisku systemu operacyjnego.

Najczęściej zawiera binarną reprezentację instrukcji konkretnego typu procesora. Oprócz tego znajdują się w nim zwykle wywołania systemowe, dlatego pliki wykonywalne zazwyczaj są specyficzne nie tylko dla danego procesora, ale też dla danego systemu operacyjnego. Nie dotyczy to sytuacji, kiedy zawierają formę pośrednią, która do uruchomienia wymaga interpretera lub maszyny wirtualnej – takie pliki mogą być zwykle uruchamiane na różnych systemach.

Specyficzną odmianą plików wykonywalnych są skrypty powłoki. Zarówno w systemie DOS jak i systemach uniksowych pliki takie można uruchamiać bezpośrednio, jak każdy inny program (np. wpisując jego nazwę w wierszu poleceń).

Zależnie od konwencji, nazwy plików wykonywalnych mogą wyróżniać się rozszerzeniem, np. w DOS i Windows przyjęte zostało rozszerzenie COM i EXE (stąd popularne określenie "egzek"). W systemach uniksowych pliki mają ustawiony atrybut wykonywalności (oznaczany literą x).

COM (COre iMage)

Struktura plików COM jest bardzo prosta, nie zawiera nagłówka ani metadanych, tylko sam kod i dane. Plik binarny ma maksymalnie 65280 (0xFF00 w systemie szesnastkowym) bajtów (nieco mniej niż 64 kB), wszystkie dane oraz kod są trzymane w jednym segmencie. Jest to dość poważną wadą, ponieważ w takim przypadku programy COM nie mogą być łączone z modułami zewnętrznymi.

Programy tego typu są ładowane przez system operacyjny zawsze pod stały adres offsetowy 0x0100, skąd są wykonywane. Nie było to problemem w przypadku ośmiobitowych maszyn, ale jest to głównym powodem dlaczego ten format wyszedł z użycia zaraz po wprowadzeniu 16-bitowych i 32-bitowych procesorów z większymi, segmentowanymi pamięciami, gdzie każdy proces dostaje swój własny segment.

Lista parametrów uruchamianego programu umieszczana jest z przesunięciem 0x0081 (ze spacją), poprzedza ją 8 bitowa długość w offsecie 0x0080 w tym samym segmencie.

MZ DOS

16-bitowy format plików wykonywalnych .EXE w systemie DOS.

Identyfikowany za pomocą dwóch znaków „MZ” (hexadecimal: 4D 5A) ASCII na początku pliku. „MZ” to inicjały nazwiska Marka Zbikowskiego, jednego z programistów systemu MS DOS. Format MZ DOS jest nowszy od COM (zastąpił COM w MS-DOS 2.0) i posiada od niego nieco większe możliwości.

Nagłówek formatu DOS MZ zawiera informacje na temat położenia poszczególnych segmentów, co pozwala na rozmieszczenie ich na dowolnych adresach w pamięci. Wspierane są również pliki wykonywalne większe niż 64 KiB.

Egzeki MZ DOS mogą być uruchamiane w DOS oraz systemach operacyjnych z rodziny Windows 9x. Na 32-bitowych systemach serii Windows NT możliwe jest ich uruchomienie za pomocą wbudowanej maszyny wirtualnej Virtual DOS. Nie można ich uruchomić na 64-bitowej wersji systemu Windows.

NS (New Executable)

16-bitowym format plików .EXE i następca formatu DOS MZ.

Używany był w systemie DOS 4.0 i w 16-bitowych wersjach Microsoft Windows (Windows 1.0, Windows 2.x i Windows 3.x).

Format ten zawiera nagłówek pliku DOS MZ (tzw. DOS stub) w celach wstecznej kompatybilności z nieobsługiwanymi wersjami DOSa.

Pliki wykonywalne formatu NE mogą być uruchamiane na 32-bitowych Windowsach. Nie są natomiast wspierane na Windowsach 64-bitowych.

PE (Portable Executable)

Format plików wykonywalnych, obiektowych oraz bibliotek dynamicznych. Używany w 32- i 64-bitowych wersjach systemów operacyjnych z rodziny Microsoft Windows. Portable oznacza "przenośny", co odnosi się do uniwersalności formatu, dostępnego w wielu architekturach systemów komputerowych.

W systemach z rodziny Windows NT, format PE jest używany m.in. przez pliki *.exe, *.dll, *.obj, *.sys (najczęściej plik sterownika urządzenia).

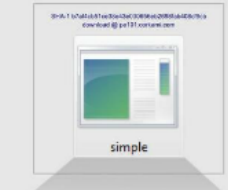
Format PE jest zmodyfikowaną wersją Uniksowego formatu COFF stąd też jego alternatywna nazwa – PE/COFF.

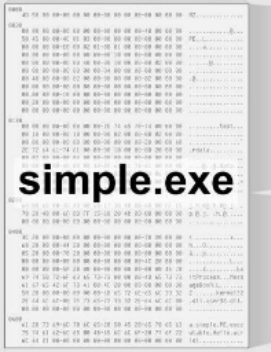
W systemach Windows NT, format PE może zawierać zarówno instrukcje z zestawu IA-32 jak i IA-64 oraz x86_64 (AMD64 i EM64T). Do wersji 4 włącznie, Windows NT (a więc de facto PE) obsługiwał również architektury MIPS, DEC Alpha i PowerPC. PE używany jest również w Microsoft Windows CE, który kontynuuje wsparcie dla kilku wariantów architektury MIPS, ARM (włączając Thumb) oraz SuperH.

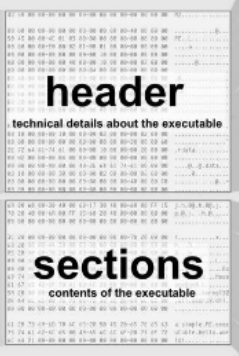
PE101 a windows executable walkthrough

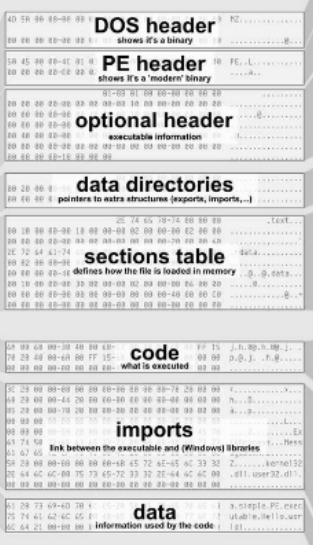
Ange Albertini
corkami.com

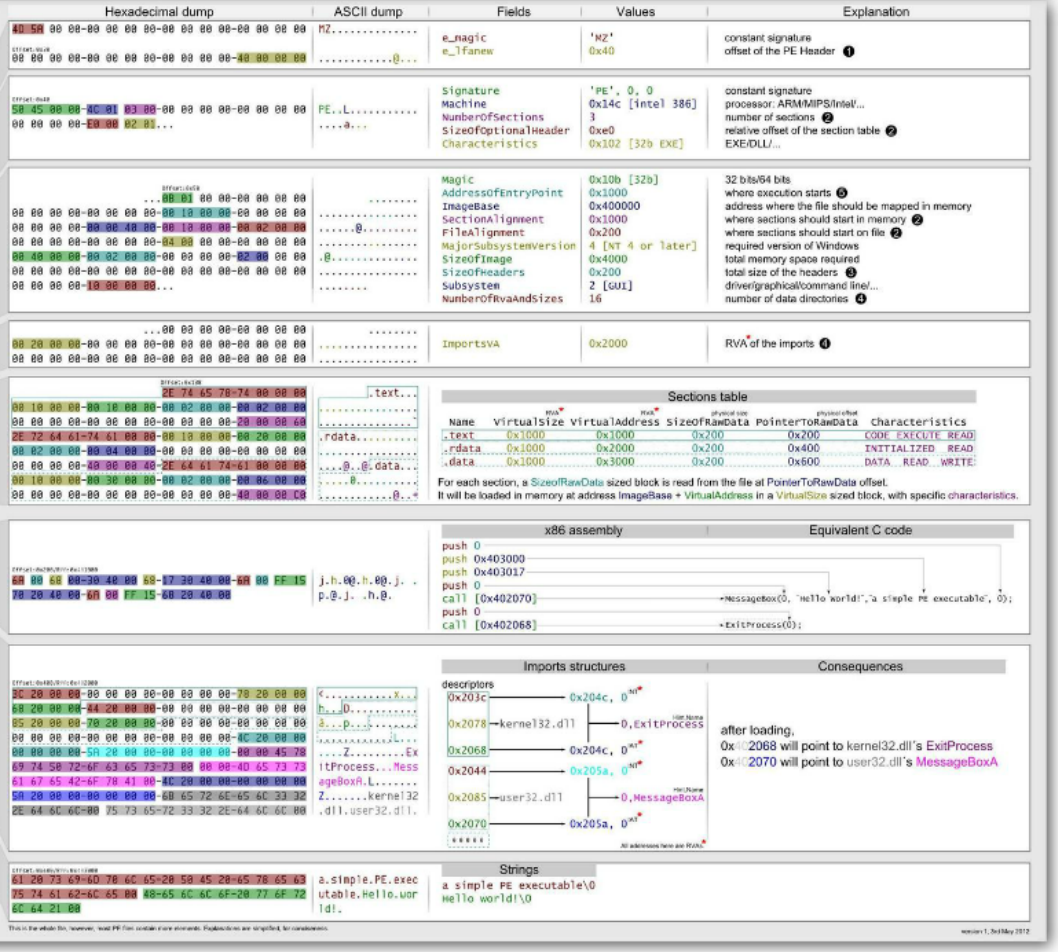
Dissected PE











Loading process

❶ Headers

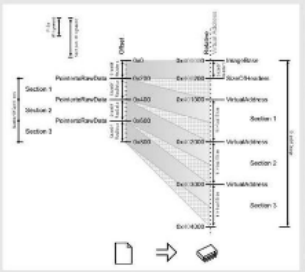
the *DOS Header* is parsed
the *PE Header* is parsed
(its offset is *DOS Header*'s e_lfanew)
the *Optional Header* is parsed
(it follows the *PE Header*)

❷ Sections table

Sections table is parsed
(it is located at: offset (*OptionalHeader*) + *SizeOfOptionalHeader*)
it contains *NumberOfSections* elements
it is checked for validity with alignments:
FileAlignments and *SectionAlignments*

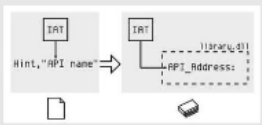
❸ Mapping

the file is mapped in memory according to:
the *ImageBase*
the *SizeOfHeaders*
the Sections table



❹ Imports

DataDirectories are parsed
they follow the *OptionalHeader*
their number is *NumOfRVAAndSizes*
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



❺ 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 (at ImageBase, RVA = 0)
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

Pliki wykonywalne w systemie Linux

Linux wspiera kilka różnych formatów plików wykonywalnych. Każdy z nich zarejestrowany jest na liście formats.

```
struct linux_binfmt {
    struct linux_binfmt * next;
    long *use_count;
    int (*load_binary)(struct linux_binprm *, struct  pt_regs * regs);
    int (*load_shlib)(int fd);
    int (*core_dump)(long signr, struct pt_regs * regs);
};
```

Każdy z formatów za pomocą funkcji register_binfmt definiuje i wstawia do listy formats elementy typu linux_binfmt z odpowiednimi dla siebie funkcjami do rozpoznawania i ładowania programu, ladowania bibliotek dzielonych i zrzutu obrazu procesu na dysk.

4 rodzaje formatów plików wykonywalnych: skrypty, format Javy, a.out, ELF

Formaty skryptów

Wykonywane są pliki tekstowe napisane w dowolnym języku skryptowym, jakiego interpreter znajduje sie w systemie.

Plik musi mieć ustawiony atrybut do wykonania. Można tego dokonać za pomoca polecenia:
chmod +x nazwa_pliku

Ścieżka interpretera zdefiniowana jest w pierwszej linijce skryptu, zaraz po znakach #!Np. #!/bin/bash/

Formaty Javy

W Linuxie istnieje możliwość bezpośredniego wywoływania plików napisanych w javie.

Spełnione muszą zostać następujące warunki:

- w systemie musi być zainstalowany Java Developers Kit for Linux
- samodzielne programy-klasy, które mają być bezpośrednio wykonywane musza mieć:
  -ustawiony atrybut do wykonania
  -zaimplementowaną metodę public void main(String args[]).
- pliki, które zawierają aplety i maja być bezpośrednio wykonywane musza mieć:
  -następujacą pierwsza linie w pliku: <!--applet-->,
  -ustawiony atrybut do wykonania.
- jeśli klasy, których używa program-klasa bądź aplet, znajdują się poza katalogiem /usr/local/java/classes/ , to należy eksportować zmienna środowiskową CLASSPATH z odpowiednio zmodyfikowanymi parametrami.

a.out

Format plików używany szczególnie w Uniksie i innych uniksopodobnych systemach operacyjnych jako format plików wykonywalnych, plików obiektowych oraz bibliotek dzielonych.

a.out jest wciąż domyślną nazwą pliku wynikowego w niektórych kompilatorach

Był to podstawowy format plików wykonywalnych w Linuxie.

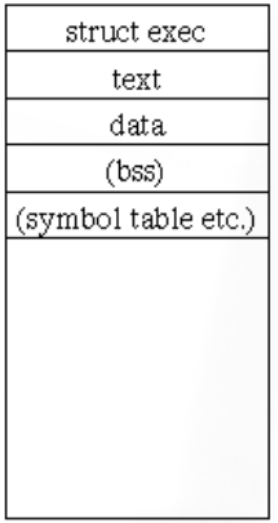
Obsługa tego formatu przez jądro zapewnia:

- ładowanie programu
- ładowanie dzielonej biblioteki procedur
- zrzut obrazu na dysk.

a.out

Plik wykonywalny w formacie a.out zawiera cztery części:

- Nagłówek podstawowy opisujący ilość sekcji w pliku
- Nagłówki sekcji
- Sekcje z kodem
- Sekcje z tablicą symboli i innymi danymi użytecznymi podczas uruchamiania programu.



a.out file format

ELF

ELF - Executable and Linking Format

Format plików binarnych stworzony przez USL (UNIX System Laboratories)

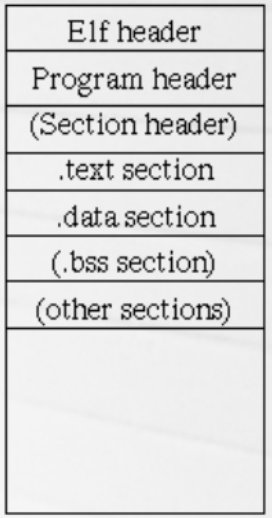
Posiada następujące możliwości:

- plik wykonywalny może być podzielony na dowolną liczbę sekcji
- sekcje maja oddzielnie określone cechy
- mogą być zaznaczone jako wcale nieładowalne do pamięci
- możliwość przeszukiwania funkcji w bibliotekach po kompilacji
- duża elastyczność i modyfikowalność
- tworzenie dzielonych bibliotek jest dużo łatwiejsze
- łatwość dynamicznego wczytywania (np. dla programów które wczytują dodatkowe moduły w trakcie działania).

ELF

Plik ELF składa się z:

- jednego nagłówka programu
- listy nagłówków programu zawierajcej zero lub wicej segmentów
- listy nagłówków sekcji zawierajcej zero lub wicej sekcji
- danych zawierajcych segmenty i sekcje



ELF file format

Porównanie formatów plików wykonywalnych

	Explicit processor declarations	Arbitrary sections	Metadata[vague]	Digital signature	String table	Symbol table	64-bit	Fat binaries	Can contain icon
a.out	No	No	No	No	Yes[1]	Yes[1]	Extension	No	No
COFF	Yes by file	Yes	No	No	Yes	Yes	Extension	No	No
ELF	Yes by file	Yes	Yes	Yes[2]	Yes	Yes[3]	Yes	Extension[4]	Extension[5]
PE	Yes by file	Yes	Yes	Yes[6]	Yes	Yes	Yes	No	Yes
Mach-O[7]	Yes by section	Some (limited to max. 256 sections)	Yes	Yes	Yes	Yes	Yes	Yes	No
SOM	Unknown	Unknown	No	No	Unknown	Yes	No	Unknown	No
Hunk	Unknown	Yes	Yes	No	No	Yes	No	Yes	No
MZ	No (x86 only)	Yes	No	No	No	No	No	No	No
DOS COM	No (x86 only)	No	No	No	No	No	No	Extension	No
CP/M-80 COM	No (8080/Z80 only)	Extension (CP/M 3 only)	No	No	No	No	No	Extension	No
CP/M-86 CMD	No (x86 only)	Yes	No	No	No	No	No	No	No
PEF[8]	Yes by file	No	No	No	Yes	Yes	No	No	No
ECOFF	Yes by file	Yes	No	No	Yes	Yes	Yes	No	No
XCOFF	Yes by file	Yes	No	No	Yes	Yes[9]	Yes	No	No
NE	Unknown	Unknown	Unknown	No	Unknown	Unknown	No	No	Yes
LX	Unknown	Unknown	Unknown	Unknown	No	Yes[10]	No	No	Yes

Formaty plików wykonywalnych dla systemu DOS, Windows i Linux

Mateusz Gorządek i Arkadiusz Grzanka