ELF[101] a Linux executable walk-through

ANGE ALBERTINI
CORKAMI.COM

DISSECTED FILE

```
~$ uname -m
armv7l
~$ ./simple.ARM
Hello World!
```

SIMPLE.ARM
SHA-1 470FA4549C36429407EF54BC1DDD7D5D2735592E
DOWNLOAD @ ELF101.CORKAMI.COM

HEADER 1/2
TECHNICAL DETAILS FOR IDENTIFICATION AND EXECUTION

ELF HEADER
IDENTIFY AS AN ELF TYPE
SPECIFY THE ARCHITECTURE

PROGRAM HEADER TABLE
EXECUTION INFORMATION

SECTIONS
CONTENTS OF THE EXECUTABLE

CODE
EXECUTABLE INFORMATION

DATA
INFORMATION USED BY THE CODE

SECTIONS' NAMES

HEADER 2/2
TECHNICAL DETAILS FOR LINKING (IGNORED FOR EXECUTION)

SECTION HEADER TABLE
LINKING (CONNECTING PROGRAM OBJECTS) INFORMATION

HEXADECIMAL DUMP / ASCII DUMP

```
7F 45 4C 46 01 01 01 00 00 00 00 00 00 00 00 00  .ELF............
02 00 28 00 01 00 00 00 60 00 00 08 40 00 00 00  ..(.....`...@...
B0 00 00 00 00 00 00 00 34 00 20 00 01 00 28 00  ........4.....(.
04 00 03 00                                      ....
```

1 ELF HEADER

FIELDS	VALUES	EXPLANATION
e_ident		
EI_MAG	0x7F, "ELF"	CONSTANT SIGNATURE
EI_CLASS, EI_DATA	1 ELFCLASS32, 1 ELFDATA2LSB	32 BITS, LITTLE-ENDIAN
EI_VERSION	1 EV_CURRENT	ALWAYS 1
e_type	2 ET_EXEC	EXECUTABLE
e_machine	28 EM_ARM	ARM PROCESSOR
e_version	1 EV_CURRENT	ALWAYS 1
e_entry	0x8000060	ADDRESS WHERE EXECUTION STARTS
e_phoff	0x40	PROGRAM HEADERS' OFFSET
e_shoff	0xB0	SECTION HEADERS' OFFSET
e_ehsize	0x34	ELF HEADER'S SIZE
e_phentsize	0x20	SIZE OF A SINGLE PROGRAM HEADER
e_phnum	1	COUNT OF PROGRAM HEADERS
e_shentsize	0x28	SIZE OF A SINGLE SECTION HEADER
e_shnum	4	COUNT OF SECTION HEADERS
e_shstrndx	3*	INDEX OF THE NAMES' SECTION IN THE TABLE

```
Offset:0x40/Address:0x8000040
01 00 00 00 00 00 00 00 00 00 00 00 08 00 00 00  ................
90 00 00 00 90 00 00 00 05 00 00 00 00 00 00 00  ................
```

2

p_type	1 PT_LOAD	THE SEGMENT SHOULD BE LOADED IN MEMORY
p_offset	0	OFFSET WHERE IT SHOULD BE READ
p_vaddr	0x8000000	VIRTUAL ADDRESS WHERE IT SHOULD BE LOADED
p_paddr	0x8000000	PHYSICAL ADDRESS WHERE IT SHOULD BE LOADED
p_filesz	0x90	SIZE ON FILE
p_memsz	0x90	SIZE IN MEMORY
p_flags	5 PF_R|PF_X	READABLE AND EXECUTABLE

```
Offset:0x60/Address:0x8000060
0D 20 A0 E3 14 10 8F E2 01 00 A0 E3 04 70 A0 E3  ..............p..
00 00 00 EF 01 00 A0 E3 01 70 A0 E3 00 00 00 EF  .........p......
```

3

ARM ASSEMBLY / EQUIVALENT C CODE

```
mov r2, #13          ; MSG_LEN
add r1, pc, #20      ; msg (pc-relative)
mov r0, #1           ; STDOUT_FILENO
mov r7, #4           ; WRITE
svc 0                ; system call          -> write(STDOUT_FILENO, "Hello World!\n", len("Hello World!\n"));

mov r0, #1           ; return code
mov r7, #1           ; SC_EXIT
svc 0                ; system call          -> exit(1);
```

```
Offset:0x88/Address:0x8000088
48 65 6C 6C 6F 20 57 6F 72 6C 64 21 0A 00        Hello.World!..
```

STRINGS

```
"Hello World!\n", 0
```

```
Offset:0x90
00 2E 73 68 73 74 72 74 61 62 00 2E 74 65 78 74  ..shstrtab..text
00 2E 72 6F 64 61 74 61 00                       ..rodata.
```

SECTION NAMES

```
"" .shstrtab .text .rodata
```

```
Offset:0xB0
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 0B 00 00 00 01 00 00 00  ................
06 00 00 00 60 00 00 08 60 00 00 00 20 00 00 00  ....`...`... ...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
11 00 00 00 01 00 00 00 02 00 00 00 80 00 00 08  ................
80 00 00 00 0D 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 01 00 00 00 03 00 00 00  ................
00 00 00 00 00 00 00 00 90 00 00 00 19 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

SECTION HEADER TABLE

INDEX	NAME	TYPE	FLAGS	ADDRESS	OFFSET	SIZE
0	<null>	0 SHT_NULL (inactive)				
1	.text	1 SHT_PROGBITS (program)	6 SHF_ALLOC (allocated) SHF_EXECINSTR (executable)	0x8000060	0x60	0x20
2	.rodata	1 SHT_PROGBITS (program)	2 SHF_ALLOC (allocated)	0x8000080	0x80	0x0D
3*	.shstrtab	3 SHT_STRTAB (string table)			0x90	0x19

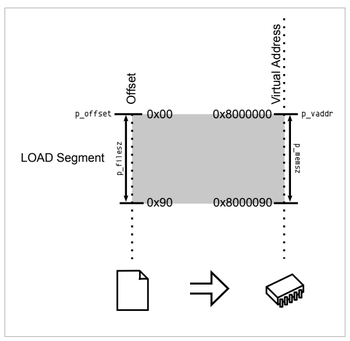
* relative offsets in names' section

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

LOADING PROCESS

1 HEADER
THE ELF HEADER IS PARSED
THE PROGRAM HEADER IS PARSED
(SECTIONS ARE NOT USED)

2 MAPPING
THE FILE IS MAPPED IN MEMORY ACCORDING TO ITS SEGMENT(S)



3 EXECUTION
ENTRY IS CALLED
SYSCALLS[KERNEL DEVELS] ARE ACCESSED VIA:
- SYSCALL NUMBER IN THE R7 REGISTER[NEW ABI]
- CALLING INSTRUCTION SVC

TRIVIA

THE ELF WAS FIRST SPECIFIED BY U.S. L.[SYSTEM V APPLICATION BINARY INTERFACE] AND U.I.[TOOL INTERFACE STANDARD]
FOR UNIX SYSTEM V, IN 1989

THE ELF IS USED, AMONG OTHERS, IN:
- LINUX, ANDROID, *BSD, SOLARIS, BEOS
- PSP, PLAYSTATION 2-4, DREAMCAST, GAMECUBE, WII
- VARIOUS OSES MADE BY SAMSUNG, ERICSSON, NOKIA,
- MICROCONTROLLERS FROM ATMEL, TEXAS INSTRUMENTS

VERSION 1.0A[NT]
2013/12/06

CC BY