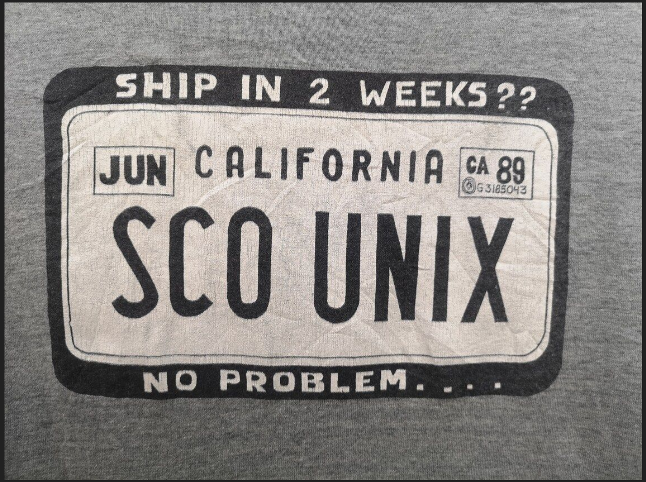
ELF: Executable and Linkable* Format

Ross Clarke

*(Extensible and Linking if you’re old enough)

History

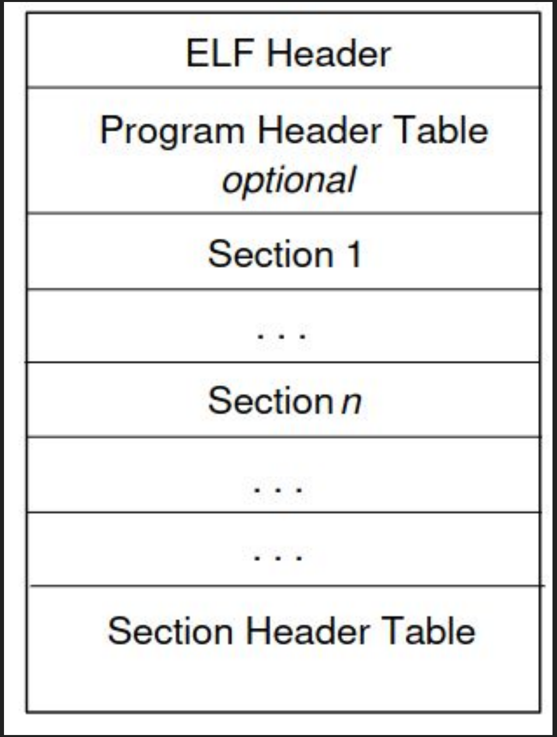
- System V Release 4.0 - APPLICATION BINARY INTERFACE
  - https://refspecs.linuxfoundation.org/elf/mipsabi.pdf (1996 version 3)
- Tool Interface Standard (TIS)
  - https://refspecs.linuxfoundation.org/elf/TIS1.1.pdf (1993)

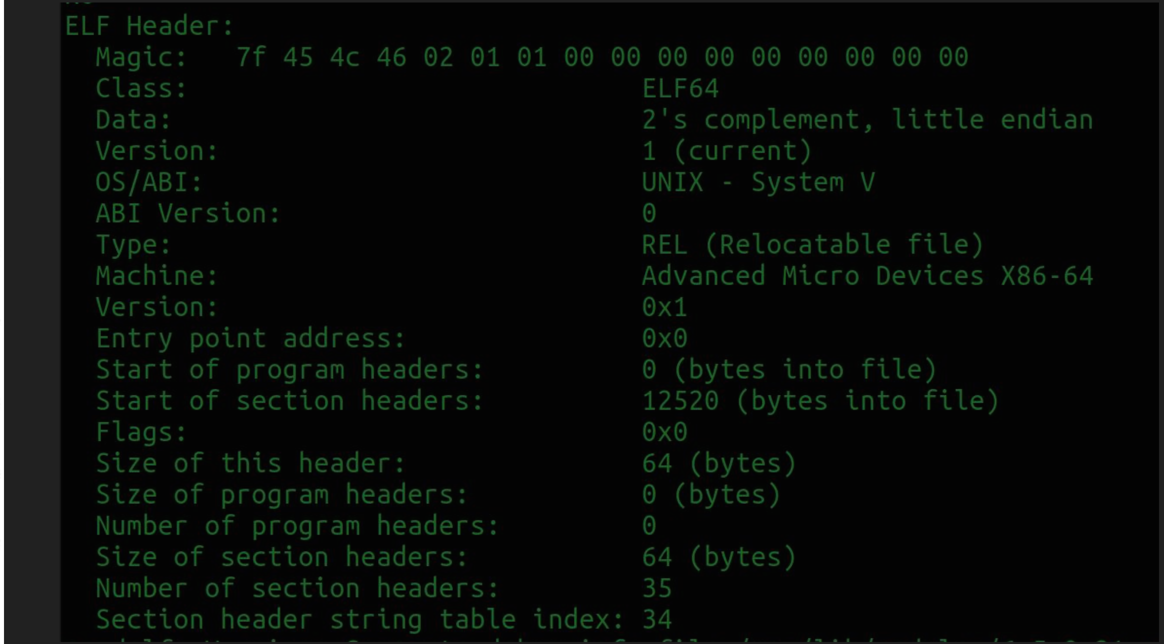


Basic Formats

- Relocatable
  - cmac.ko
- Executable
  - Any C program compiled with `gcc –no-pie`
- Shared Object
  - ls

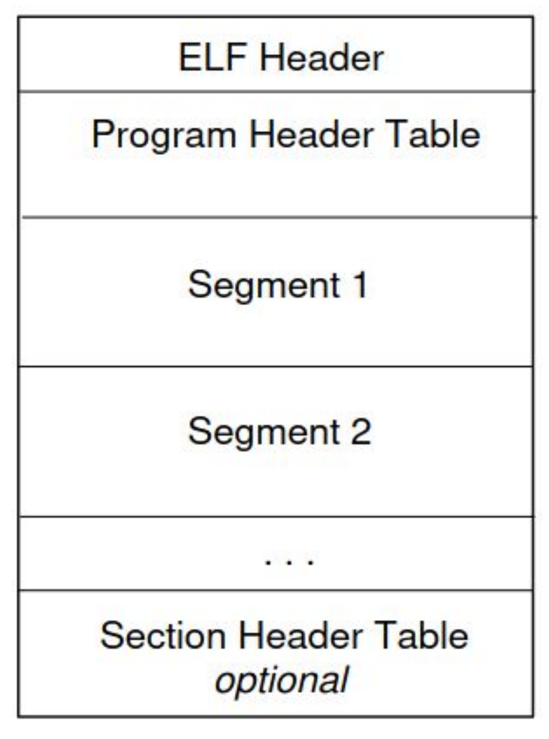
Relocatable





```
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              REL (Relocatable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x0
  Start of program headers:          0 (bytes into file)
  Start of section headers:          12520 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           0 (bytes)
  Number of program headers:         0
  Size of section headers:           64 (bytes)
  Number of section headers:         35
  Section header string table index: 34
```

Executable



ELF Header

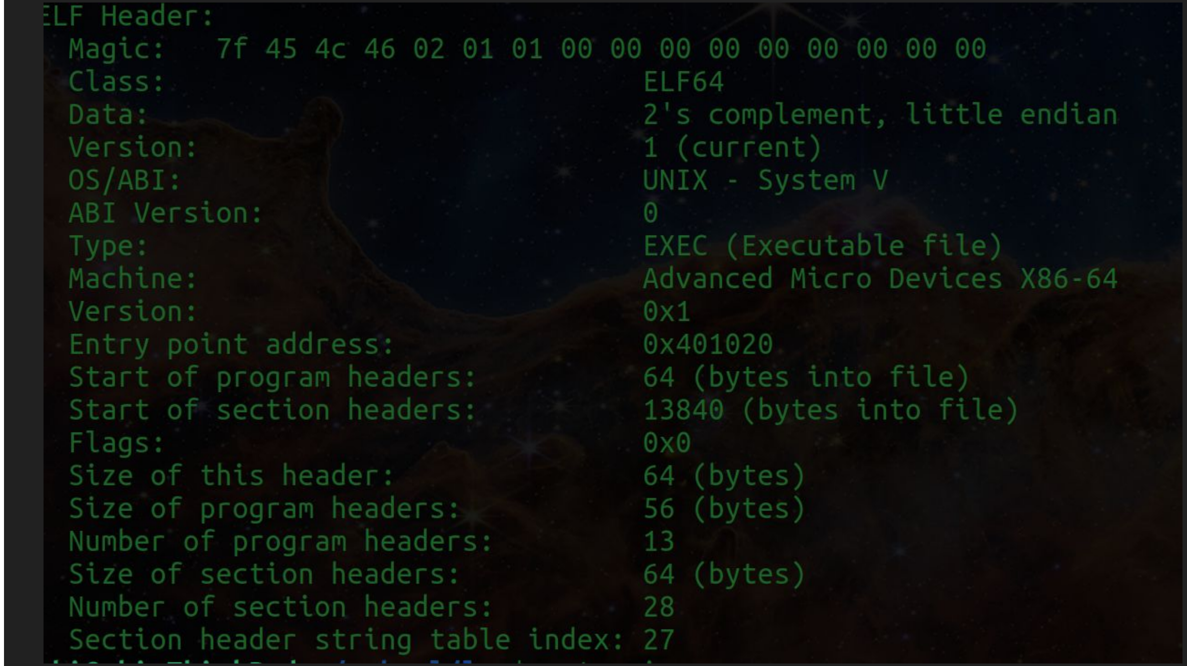
Program Header Table

Segment 1

Segment 2

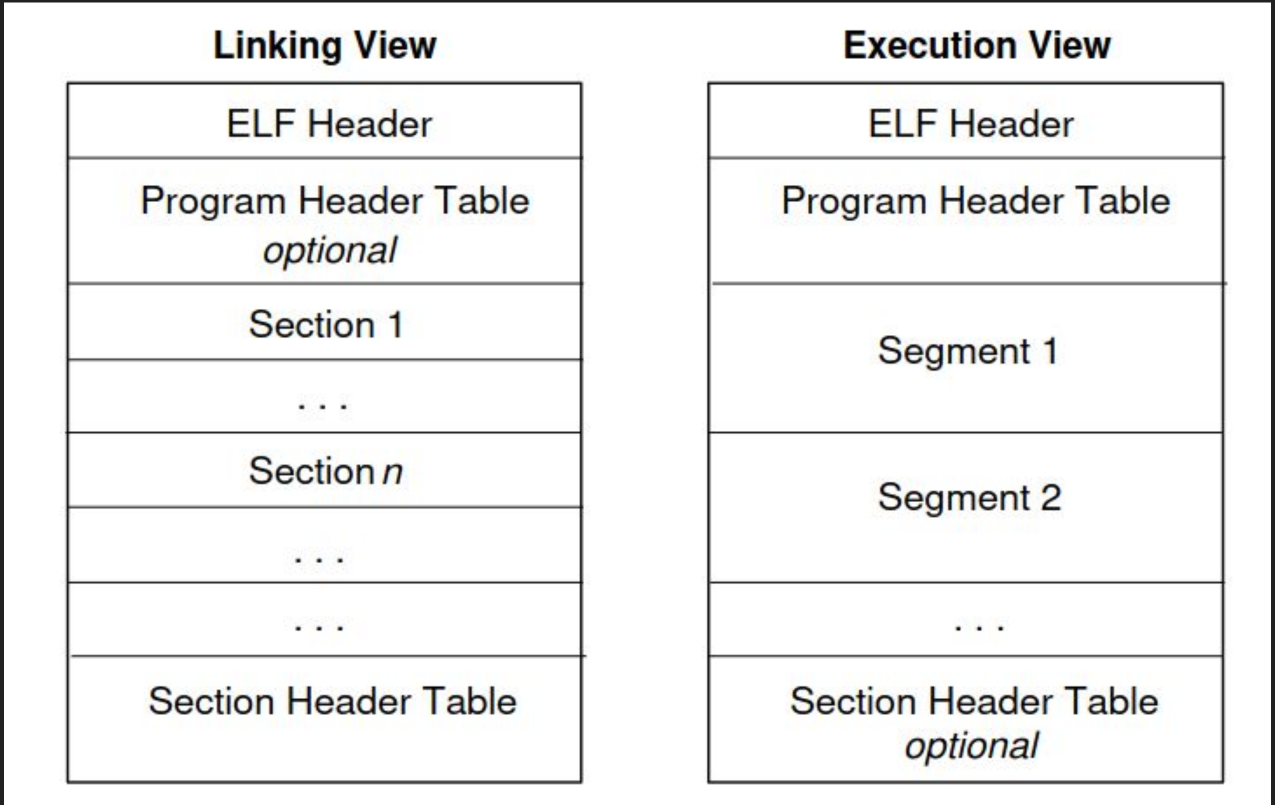
. . .

Section Header Table
optional



```
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x401020
  Start of program headers:          64 (bytes into file)
  Start of section headers:          13840 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         13
  Size of section headers:           64 (bytes)
  Number of section headers:         28
  Section header string table index: 27
```

Shared Object



Linking View

ELF Header

Program Header Table
optional

Section 1

. . .

Section *n*

. . .

. . .

Section Header Table

Execution View

ELF Header

Program Header Table

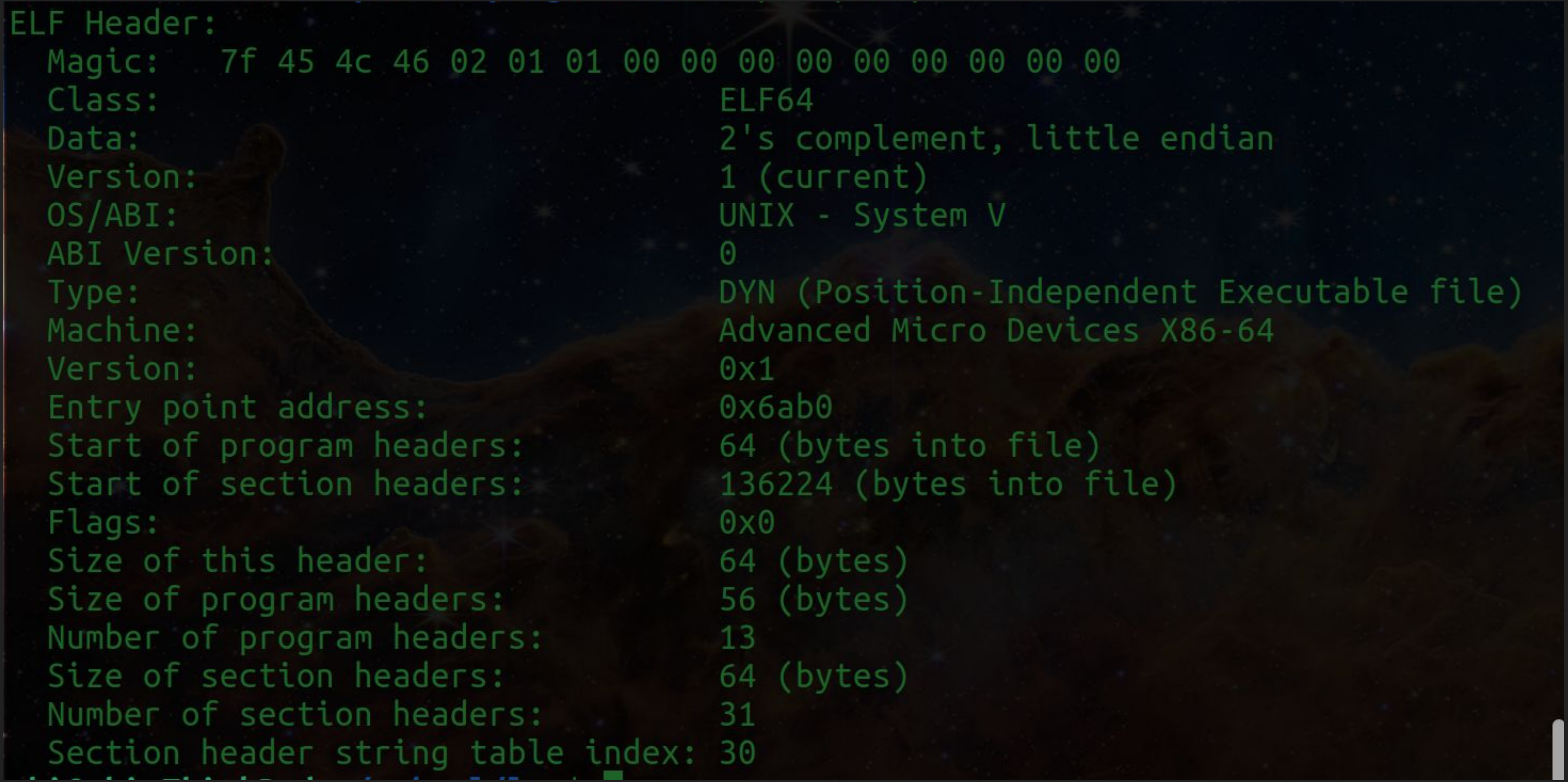
Segment 1

Segment 2

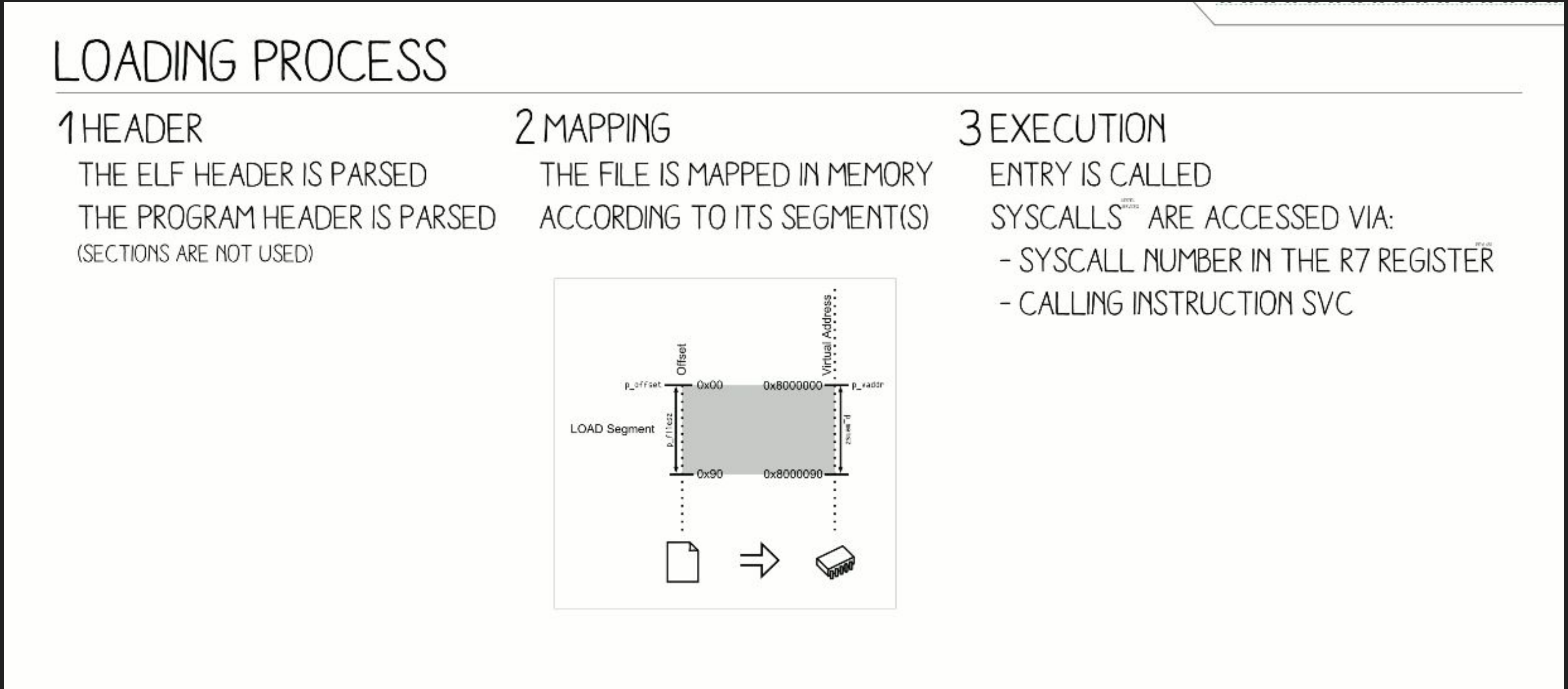
. . .

Section Header Table
optional

Shared Object



```
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Position-Independent Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x6ab0
  Start of program headers:          64 (bytes into file)
  Start of section headers:          136224 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         13
  Size of section headers:           64 (bytes)
  Number of section headers:         31
  Section header string table index: 30
```



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)

Offset
Virtual Address
p_offset
0x00
0x8000000
p_vaddr
LOAD Segment
p_filesz
p_memsz
0x90
0x8000090

3 EXECUTION

ENTRY IS CALLED

SYSCALLS ARE ACCESSED VIA:

- SYSCALL NUMBER IN THE R7 REGISTER
- CALLING INSTRUCTION SVC

https://github.com/corkami

Tools!

- readelf
- ldd
- objdump
- strace
- lsof