Lecture 03: Linking and Loading of an ELF File

Vivek Kumar Computer Science and Engineering IIIT Delhi vivekk@iiitd.ac.in



Today's Class

- Revisiting Executable File Format (ELF)
- Dissecting the sections of an ELF
 - o Symbol resolution

Class Quiz (1/4)

 Match the following structures with their respective names and responsibilities

| (A) Elf32_Phdr | (X) ELF header | (1) Information for Loader |
|----------------|------------------------|----------------------------|
| (B) Elf32_Ehdr | (Y) ELF Section Header | (2) Stores roadmap of ELF |
| (C) Elf32_Shdr | (Z) ELF Program Header | (3) Information for Linker |

Class Quiz (2/4)

- What is the meaning of "entrypoint" address in an executable?
 - 1. Offset of **SHDR** table from the beginning of ELF file
 - 2. Offset of **PHDR** table from the beginning of ELF file
 - 3. Address of "main" method
 - 4. Address of "_start" method
- Where it is stored inside the ELF binaries?
 - 1. Program header table
 - 2. Section header table
 - 3. ELF header
 - 4. Segments
 - 5. Sections

Class Quiz (3/4)

- Which of these are required at the compile time and runtime, respectively?
 - 1. ELF header
 - 2. PHDR
 - 3. SHDR
 - 4. Segments
 - 5. Sections

Class Quiz (4/4)

 Match the location shown with their respective section name inside the ELF file





Summary: Portability with ELF

- Recall, one of the challenges of OS is to support portability
- Machine code is architecture dependent
- Having an architecture independent layout of object files helps in achieving portability
 - The interpreter (loader + dynamic linker) becomes portable, and could have same set of steps to load executables across different architectures (portability)
 - Easier to change and test the tools used for manipulating object files



Today's Class

- Revisiting Executable File Format (ELF)
- Dissecting the sections of an ELF
 - o Symbol resolution

How Linker Reads Section Names?

- The section header does not contains the name of its corresponding section
 - It's structure has a member that contains an offset into the ELF file where the name of this section is stored
- The names of all the sections are stored inside .shstrtab section
- Each section header stores the offset into the .shstrtab section (not the SHDR but the actual section) where the name of that section is stored as a null terminated string

Motivation Example It is the same motivational example we discussed in Document header containing Submission-1 Lecture 02 section summary Changes shown in orange and red color Ο numResearchAreas=3 researchAreasInfoOffset=56 Summary for 3 We added a new section research papers shstrndx=2 "S" that contains the Summary-1 in category-A name of all the sections in Summary-2 Summary-3 this document Summarv-1 Summary for Document header contains 0 Summary-2 the index of this particular section summary 5 research Summary-3 papers in Summary-4 For simplicity, assume category-B Ο Summary-5 each slot shown in figure Α is 4 bytes (e.g., header is В of 3x4 bytes) Section containing names Each section summary nameOffset=0 of each section numSummaries=3 now also contains offset summaryOffset=12 (bytes) from the start of the content of section "S" nameOffset=4 numSummaries=5 where the name is stored summaryOffset=24 nameOffset=8 numSummaries=3 summaryOffset=44 CSE231: Operating System © Vivek Kumar 12

Lecture 02: Linking and Loading of an ELF file

The ".shstrtab" Section in ELF

| vivek@ | possur | <pre>n:~/elf_os-l2\$ readelf -p .shstrtab</pre> |
|--------|------------|---|
| 0+ | - I | |
| String | aump | of section '.shstrtab': |
| | 11 | . Symtab |
| L | 9] | strtad |
| L | 11] | .snstrtab |
| L | 16] | .interp |
| L | 23] | .note.ABI-tag |
|] | 31] | .note.gnu.build-id |
|] | 44] | .gnu.hash |
| [| 4e] | .dynsym |
| Ι | 56] | .dynstr |
| [| 5e] | .gnu.version |
| Γ | 6b] | .gnu.version_r |
| Γ | 7a] | .rel.dyn |
| Γ | 83] | .rel.plt |
| Γ | 8c] | .init |
| Γ | 92] | .plt.got |
| Γ | 9b] | .text |
| [| a1] | .fini |
| Γ | a7] | .rodata |
| Γ | af] | .eh_frame_hdr |
| Γ | bd] | .eh_frame |
| Γ | c7] | .init_array |
| Γ | d3] | .fini_array |
| Γ | df] | .dynamic |
| Γ | e8] | .data |
| I | ee] | .bss |
| Г | f31 | .comment |
| | | |



- It is a String Table Section
- Stores the name of each sections in the binary
 - String (NULL terminated)
- As with other sections, this section is also a contiguous chunk of memory

First column is the index (hex value) of the first character of that section name. First entry starts at index 1

CSE231: Operating Systems

Lecture 02: Linking and Loading of an ELF file

Reading ".shstrtab" in Plain C

| | ELF Header: Magic: 7f 45 4c 46 01 01 01 00 0 Class: Data: Version: OS/ABI: ABI Version: Type: Machine: Version: Entry point address: Start of program headers: Start of section headers: Flags: Size of this header: Size of program headers: Size of section headers: Size of section headers: Section header_string table index: Section headers: Section headers: Section headers: Section header_string table index: Section headers: Section header_string table index: Section headers: Section head | 200 00 00 00 00 00 00 00 00 ELF32 2's complement, little er 1 (current) UNIX - System V 0 REL (Relocatable file) Intel 80386 0×1 0×0 0 (bytes into file) 860 (bytes into file) 860 (bytes) 0 40 (bytes) 14 13 Addr Off Size 0000000 00034 00008 0000000 00035 00007 0000000 00005 00007 0000000 00005 00007 0000000 00005 00078 0000000 00074 00003 | <pre>#define EI_NIDENT 16 typedef struct { unsigned char Elf32_Half Elf32_Word Elf32_Off Elf32_Word Elf32_Half Elf32_Half Elf32_Half Elf32_Half Elf32_Half Elf32_Half Elf32_Half Elf32_Half Elf32_Ehdr; typedef struct { Elf32_Word Elf32_Word Elf32_Off Elf32_Word Elf32_Word</pre> | <pre>e_ident[EI_NIDENT]; e_type; e_machine; e_version; e_entry; e_phoff; e_flags; e_ehsize; e_phentsize; e_shentsize; e_shentsize; e_shentsize; e_shstrndx; e_shstrndx; sh_type; sh_flags; sh_addr; sh_size; sh_link; sh_info; sh_addralign; sh_entsize;</pre> | 1. 2. 3. 4. 5. | Open the a.out in O_RDONLY mode Read the value of the following member in the Elf32_Ehdr a. e_shoff b. e_shentsize (fixed) c. e_shstrndx Move to the offset e_shoff + e_shstrndx* e_shentsize a. Starting address of Elf32_Shdr for ".shstrtab" Read the value of the following members in the Elf32_Shdr a. sh_offset b. sh_size Move to the offset sh_offset from the start of the file a.out a. Content of the ".shstrtab" starts from this address b. Total content size "sh_size" bytes |
|-----|--|---|--|--|----------------------------|---|
| CSE | 231: Operating Systems | | © Vivek | Kumar | | 14 |

14

Lecture 02: Linking and Loading of an ELF file

Reading Section Names in Plain C

}

| Magic: /T 45 4C 46 0 | 1 01 01 00 0 | 0 00 00 00 0 | 0 00 00 00 | | |
|---|--|---|--|--|--|
| Class: | | ELF32 | | | |
| Data: 2's complement, | | | ent, little er | | |
| Version: | 1 (current) | | | | |
| OS/ABI: | UNTX - System V | | | | |
| ABI Version: | 9 | | | | |
| Type: | BEL (Belocatable file) | | | | |
| Machinat | Intel 80386 | | | | |
| Machine: | Machine: | | | | |
| Version: | 0.0 | | | | |
| Entry point address: | Entry point address: | | | | |
| Start of program heade | rs: | 0 (bytes into file) | | | |
| Start of section heade | rs: | 860 (bytes | into file) | | |
| Flags: | | 0x0 | | | |
| Size of this header: | | 52 (bytes) | | | |
| Size of program header | s: | 0 (bytes) | | | |
| Number of program head | ers: | 0 | | | |
| Size of section header | s: | 40 (bytes) | | | |
| Number of section head | ers: | 14 | | | |
| Section header string | table index: | 13 | | | |
| | | | | | |
| | | | | | |
| text | | | | | |
| | | | | | |
| hcc | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| als stute ls | _ | _ | | | |
| .shstrtab | | | | | |
| .shstrtab | | | | | |
| .shstrtab | | | | | |
| .shstrtab | - | | | | |
| .shstrtab | | | | | |
| .shstrtab | | | | | |
| .shstrtab | Туре | Addr 0 | ff Size | | |
| Section Headers: | Type NULL | Addr 0 90000000 9 | ff Size 00000 000000 | | |
| .shstrtab | Type NULL GROUP | Addr 0 00000000 0 0000000 0 | ff Size 00000 000000 00034 000008 | | |
| Section Headers: [Nr] Name [0] [1] .group [2] .text | Type NULL GROUP PROGBITS | Addr 0 00000000 00000000 00000000 00000000 | ff Size 00000 000000 00034 000008 00034 00008 | | |
| Section Headers: [NT] Name [0] [1] .group [2] .text [3] .rel.text | Type NULL GROUP PROGBITS REL | Addr O 00000000 0 00000000 0 00000000 0 000000 | ff Size 90000 00000 90034 000086 9003c 00008f 90220 00008f 90220 00008 | | |
| Section Headers: [Nr] Name [0] [1] .group [2] .text [3] .rel.text [4] .data | Type NULL GROUP PROGBITS REL PROGBITS MODITS | Addr 0 00000000 00000000 00000000 00000000 0000 | ff Size 00000 000000 00034 000008 0003c 00008f 00220 000035 0002b 000000 | | |
| .shstrtab Section Headers: [Nr] Name [0] [1] .group [2] .text [3] .rel.text [4] .data [5] .bss | Type NULL GROUP PROGBITS REL PROGBITS NOBITS NOBITS | Addr 0 00000000 0 00000000 0 00000000 0 000000 | ff Size 00000 00000 00034 000008 00034 00008 00290 000038 00020 000038 00020 000008 00020 000000 | | |
| Section Headers: [Nr] Name [0] [1].group [2].text [3].rel.text [4].data [5].bss [6].textx86.get_P [7].comment | Type NULL GROUP PROCBITS REL PROGBITS NOBITS PROCBITS PROCBITS | Addr O 00000000 0 0000000 0 0000000 0 0000000 0 000000 | ff Size 00000 000000 00034 000008 0003c 00008f 00220 00008f 00220 00008 000cb 000000 000cb 000000 000cb 000004 | | |
| Section Headers: [NT] Name [0] [1] .group [2] .text [3] .rel.text [4] .data [5] .bss [6] .textx86.get_p [7] .comment [8] .prete @MLstack | Type NULL GROUP PROGBITS REL PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS | Addr 0 0 0000000 0 000000 0 000000 0 000000 0 0000000 0 00000000 | ff Size 00000 000000 0034 000008 0003c 00008 00220 00008 000cb 000000 000cb 000000 000cb 000000 000cb 000004 000cc 00002a | | |
| Section Headers: [Nr] Name [0] [1] .group [2] .text [3] .rel.text [4] .data [5] .bss [6] .textx86.get_P [7] .comment [8] .note.0NU-stack [9] eh frame | Type NULL GROUP PROGBITS REL PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS | Addr 0 00000000 0 00000000 0 00000000 0 000000 | ff Size 00000 000000 00034 000008 00034 000088 00030 00008 00020 000008 00020 000000 00020 000000 00020 000000 00020 000000 00040 000000 | | |
| Section Headers: [NT] Name [0] [1].group [2].text [3].rel.text [4].data [5].bss [6].textx86.get_p [7].comment [8].note.GNU-stack [9].eh_frame [10] | Type NULL GROUP PROGBITS REL PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS | Addr 0 00000000 0 00000000 0 00000000 0 000000 | ff Size 00000 000000 00034 000008 00032 000081 00032 000081 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000004 00040 000040 000400 000040 000400 000040 000400 000400 000400 000400 000400 000400 000400 000400 0004000 000400 | | |
| Section Headers: [NT] Name [0] [1] .group [2] .text [3] .rel.text [4] .data [5] .comment [6] .rete.GNU-stack [9] .eh_frame [10] .rel.eh_frame | Type NULL GROUP PROGBITS REL PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS REL SYMTAR | Addr 0 00000000 00000000 00000000 00000000 0000 | ff Size 00000 000000 00034 000008 0003c 00008f 00200 000038 000cb 000000 000cb 000000 000cb 000004 000cf 00002a 000f9 000008 000f9 000008 000f9 00008 000f9 00008 000f9 00008 | | |
| Section Headers: [Nr] Name [0] [1] .group [2] .text [3] .rel.text [4] .data [5] .bss [6] .textx86.get_p [7] .comment [8] .note.GNU-stack [9] .eh_frame [10] .rel.eh_frame [11] .symtab [12] .strtab | Type NULL GROUP PROGBITS REL PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS REL SYMTAB STRTAB | Addr 0 00000000 0 00000000 0 00000000 0 000000 | ff Size 00000 000000 00034 000085 00290 000038 00020 000004 00020 000004 00020 000004 00020 000004 00020 000004 00020 000004 00020 000004 00020 000078 00072 000078 00022 000018 00024 00002 | | |
| Section Headers: [NT] Name [0] [1].group [2].text [3].rel.text [4].data [5].bss [6].textx86.get_p [7].comment [8].note.GNU-stack [9].eh_frame [10].rel.eh_frame [11].symtab [12].strtab [13].shstrtab | Type NULL GROUP PROGBITS REL PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS STRTAB STRTAB | Addr 0 00000000 0 0000000 0 0000000 0 000000 | ff Size 00000 000000 00034 00008 0003c 00008 0003c 00008 0003c 00008 0004c 000000 000cb 000004 000cf 00002a 000cf 000048 000cf 000004 000cf 000008 000cf 000008 000cf 000000 000cf 000018 00174 000000 002c0 000007a | | |
| Section Headers: [Nr] Name [0] [1] .group [2] .text [3] .rel.text [4] .data [5] .bss [6] .textx86.get_p [7] .comment [8] .note.0NU-stack [9] .eh_frame [10] .rel.eh_frame [11] .syntab [12] .strtab [13] .shstrtab | Type NULL GROUP PROGBITS REL PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS PROGBITS REL SYMTAB STRTAB STRTAB | Addr O 00000000 0 00000000 0 00000000 0 000000 | ff Size 00000 000000 00034 000081 00032 000081 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00040 000000 00174 000000 00254 0000000 00254 0000000 00254 0000000 | | |

| #define | EI_NIDENT 16 | |
|---------|---------------|--------------------------------|
| typedef | struct { | |
| | unsigned char | <pre>e_ident[EI_NIDENT];</pre> |
| | Elf32_Half | e_type; |
| | Elf32_Half | e_machine; |
| | Elf32_Word | e_version; |
| | Elf32_Addr | e_entry; |
| | Elf32_Off | e_phoff; |
| | Elf32_Off | e_shoff; |
| | Elf32_Word | e_flags; |
| | Elf32_Half | e_ehsize; |
| | Elf32_Half | <pre>e_phentsize;</pre> |
| | Elf32_Half | e_phnum; |
| | Elf32_Half | <pre>e_shentsize;</pre> |
| | Elf32_Half | e_shnum; |
| | Elf32_Half | <pre>e_shstrndx;</pre> |
| } Elf32 | Ehdr; | |

| typedef | struct { | |
|---------|------------|--------------------------|
| | Elf32 Word | sh name; |
| | Elf32_Word | sh_type; |
| | Elf32 Word | sh flags; |
| | Elf32_Addr | sh_addr; |
| | Elf32_Off | <pre>sh_offset;</pre> |
| | Elf32_Word | sh_size; |
| | Elf32_Word | <pre>sh_link;</pre> |
| | Elf32_Word | <pre>sh_info;</pre> |
| | Elf32_Word | <pre>sh_addralign;</pre> |
| | Elf32_Word | <pre>sh_entsize;</pre> |
| } Elf32 | Shdr; | |

Iterate through all entries in the SHDR 6. one by one

- SHDR array starts at Ο e shoff
- 7. Find the value of sh name in the SHDR entry
- Name of this section 8. will be inside the content of the section ".shstrtab and starting at the location sh name
 - printf (%s) at that Ο location prints the name

CSE231: Operating Systems

Relocatable file fib.o

Linker: Merges the Sections



 Relocation is the process of merging the sections and resolving the symbol addresses

Linker: Relocation



Reference Materials

- Executable and Linkable Format
 - o https://man7.org/linux/man-pages/man5/elf.5.html
 - o https://www.sco.com/developers/devspecs/gabi41.pdf
 - o <u>https://wiki.osdev.org/ELF_Tutorial</u>

