Lecture 03: Linking and Loading of an ELF File

Vivek Kumar

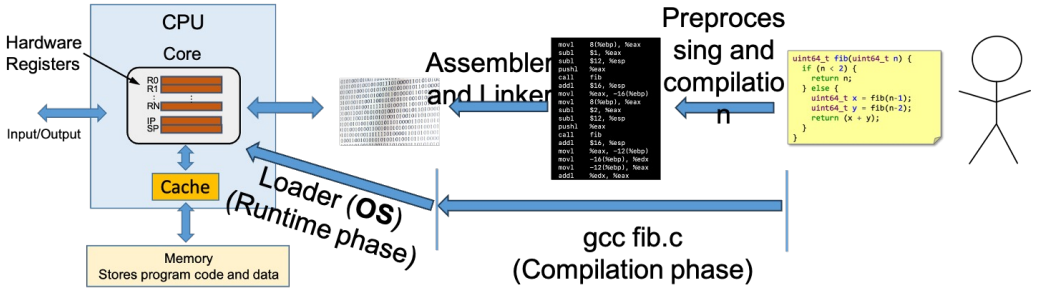
Computer Science and Engineering

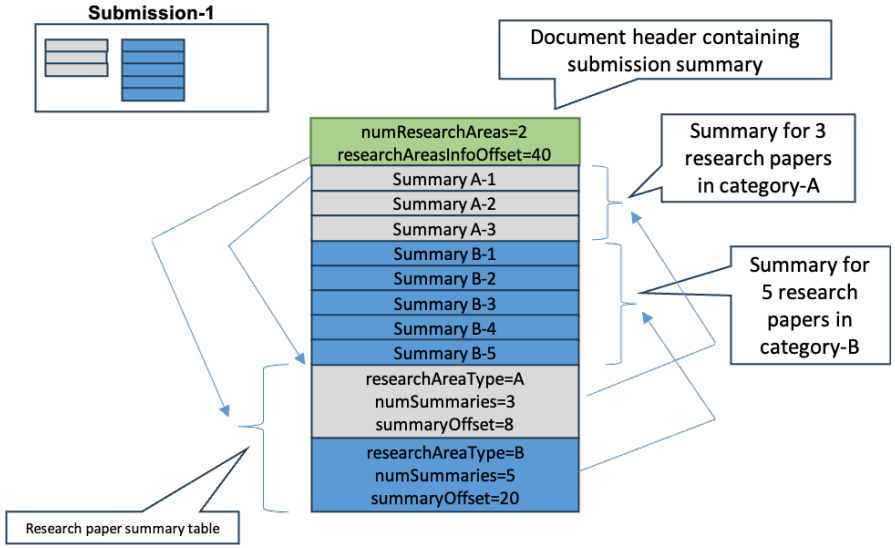
IIIT Delhi

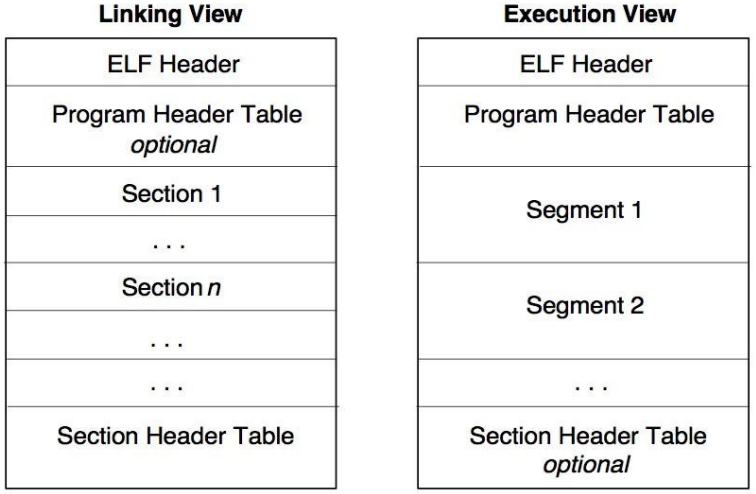
vivekk@iiitd.ac.in

Last Lecture

```
$ gcc -static -o fib fib.c
$ file fib
...ELF 64-bit executable...statically linked
$ ls -l fib
-rwxrwxr-x 1 vivek vivek 845304 Aug  6 10:26 fib
$ gcc -o fib fib.c
$ ls -l fib
-rwxrwxr-x 1 vivek vivek 8328 Aug  6 10:27 fib
```







- OS should be able to execute a variety of user applications requiring various other libraries (e.g., printf is provided by glibc library)
  - Although programs could differ, but once compiled they are stored as a file in a standard format on a given OS (e.g. ELF on Linux)
  - It makes the job of linker and loader quite easy

Today's Class

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

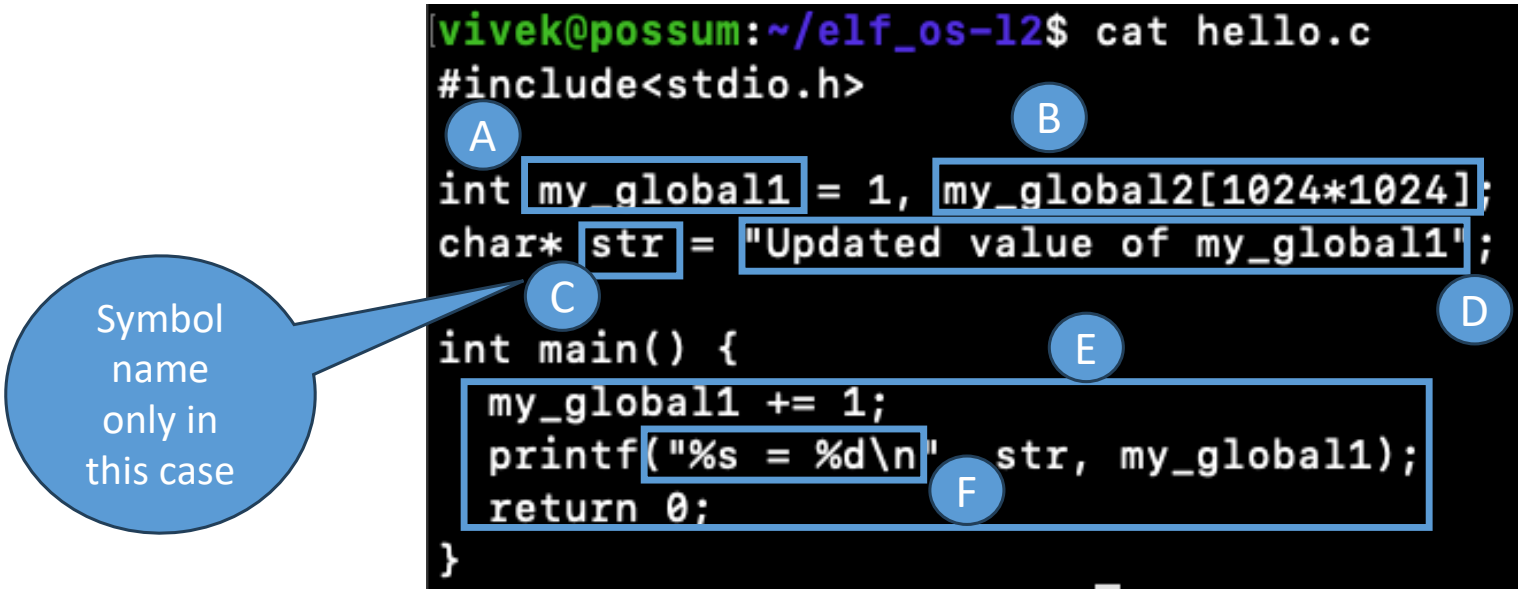
- Where it is stored inside the ELF binaries?
  1. Program header table
  2. Section header table
  3. ELF header
  4. Segments
  5. Sections

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



1. .bss
2. .strtab
3. .rodata
4. .text
5. .data

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

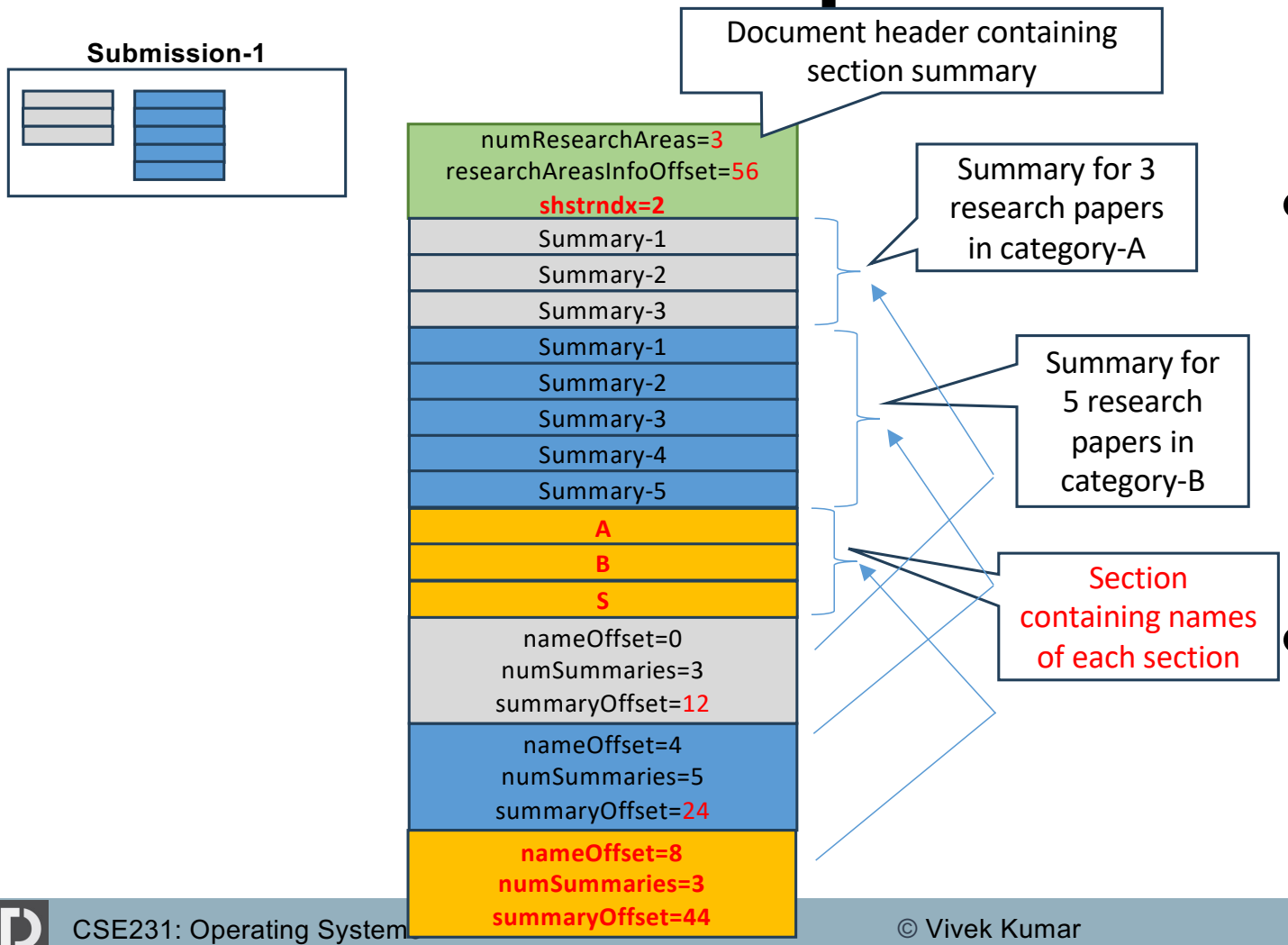
Today's Class

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

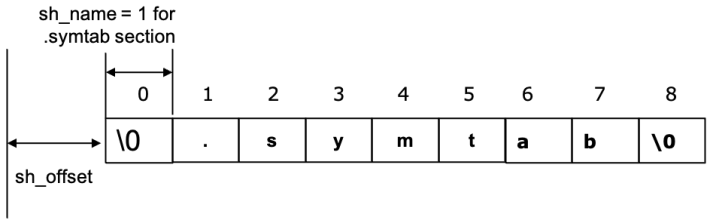
Motivation Example



- It is the same motivational example we discussed in Lecture 02
  - Changes shown in **orange** and **red** color
- We added a new section "S" that contains the name of all the sections in this document
  - Document header contains the index of this particular section summary
  - For simplicity, assume each slot shown in figure is 4 bytes (e.g., header is of 3x4 bytes)
- Each section summary now also contains offset (bytes) from the start of the content of section "S", where the name is stored

The ".shstrtab" Section in ELF

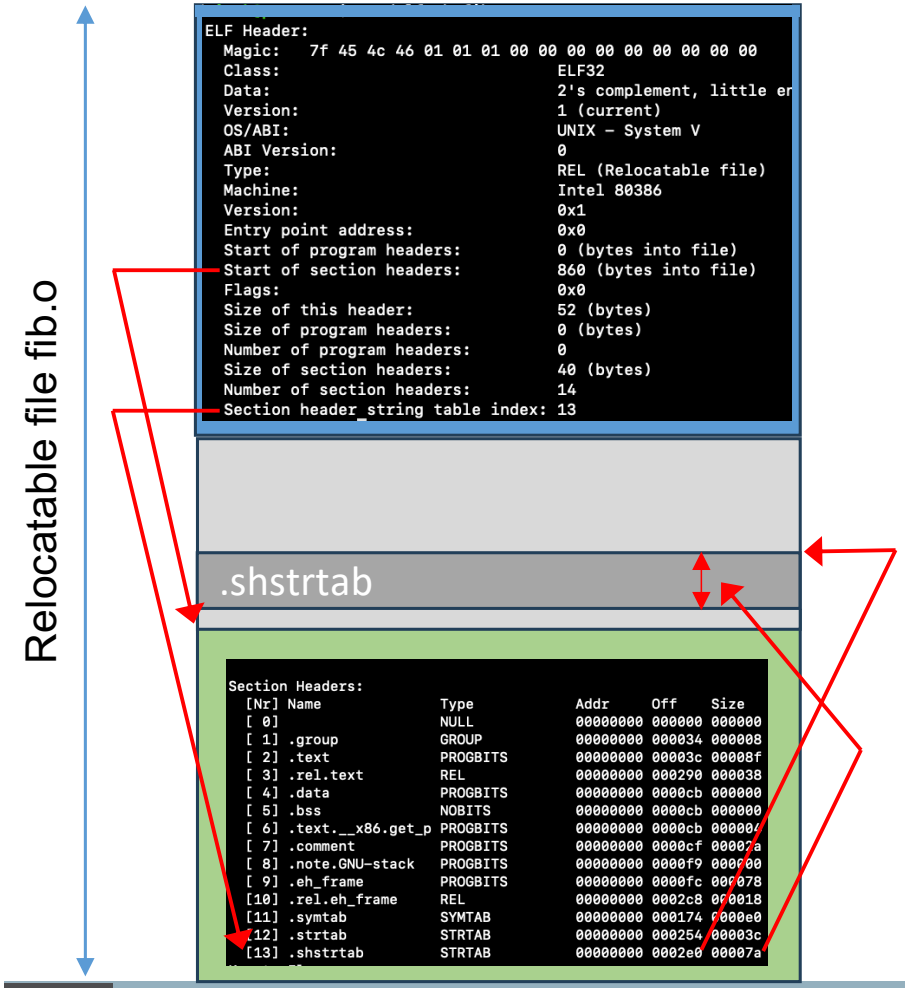
```
vivek@possum:~/elf_os-12$ readelf -p .shstrtab hello

String dump of section '.shstrtab':
  [     1]  .symtab
  [     9]  .strtab
  [    11]  .shstrtab
  [    1b]  .interp
  [    23]  .note.ABI-tag
  [    31]  .note.gnu.build-id
  [    44]  .gnu.hash
  [    4e]  .dynsym
  [    56]  .dynstr
  [    5e]  .gnu.version
  [    6b]  .gnu.version_r
  [    7a]  .rel.dyn
  [    83]  .rel.plt
  [    8c]  .init
  [    92]  .plt.got
  [    9b]  .text
  [    a1]  .fini
  [    a7]  .rodata
  [    af]  .eh_frame_hdr
  [    bd]  .eh_frame
  [    c7]  .init_array
  [    d3]  .fini_array
  [    df]  .dynamic
  [    e8]  .data
  [    ee]  .bss
  [    f3]  .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

Reading ".shstrtab" in Plain C

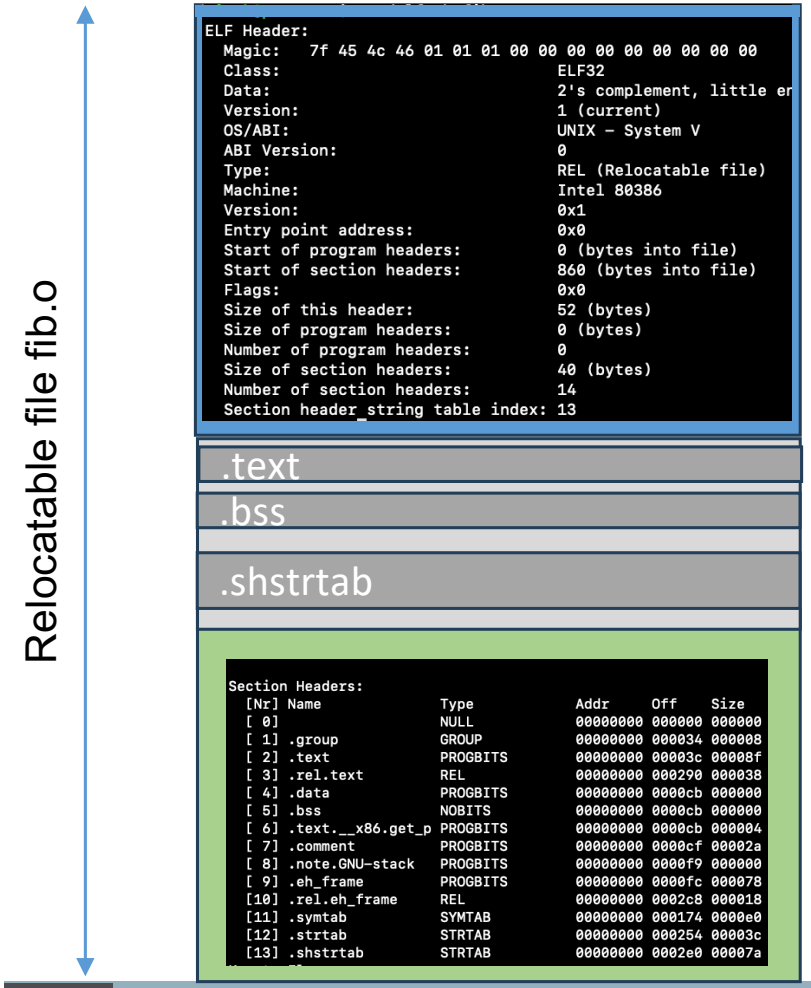



```
#define EI_NIDENT 16

typedef struct {
        unsigned char   e_ident[EI_NIDENT];
        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      e_phentsize;
        Elf32_Half      e_phnum;
        Elf32_Half      e_shentsize;
        Elf32_Half      e_shnum;
        Elf32_Half      e_shstrndx;
} Elf32_Ehdr;
```

```
typedef struct {
        Elf32_Word      sh_name;
        Elf32_Word      sh_type;
        Elf32_Word      sh_flags;
        Elf32_Addr      sh_addr;
        Elf32_Off       sh_offset;
        Elf32_Word      sh_size;
        Elf32_Word      sh_link;
        Elf32_Word      sh_info;
        Elf32_Word      sh_addralign;
        Elf32_Word      sh_entsize;
} Elf32_Shdr;
```

1. Open the a.out in O_RDONLY mode
2. Read the value of the following member in the Elf32_Ehdr
   a. e_shoff
   b. e_shentsize (fixed)
   c. e_shstrndx
3. Move to the offset e_shoff + e_shstrndx* e_shentsize
   a. Starting address of Elf32_Shdr for ".shstrtab"
4. Read the value of the following members in the Elf32_Shdr
   a. sh_offset
   b. sh_size
5. 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

Reading Section Names in Plain C

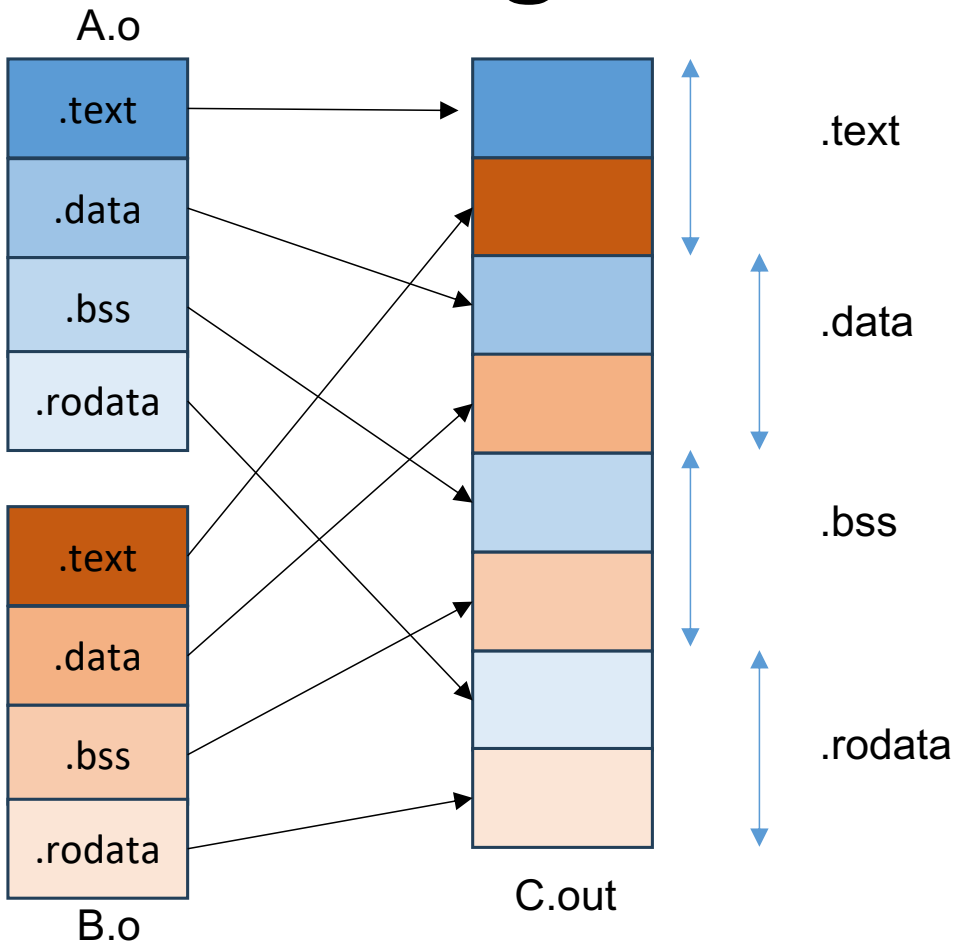



```
#define EI_NIDENT 16

typedef struct {
        unsigned char   e_ident[EI_NIDENT];
        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      e_phentsize;
        Elf32_Half      e_phnum;
        Elf32_Half      e_shentsize;
        Elf32_Half      e_shnum;
        Elf32_Half      e_shstrndx;
} Elf32_Ehdr;
```

```
typedef struct {
        Elf32_Word      sh_name;
        Elf32_Word      sh_type;
        Elf32_Word      sh_flags;
        Elf32_Addr      sh_addr;
        Elf32_Off       sh_offset;
        Elf32_Word      sh_size;
        Elf32_Word      sh_link;
        Elf32_Word      sh_info;
        Elf32_Word      sh_addralign;
        Elf32_Word      sh_entsize;
} Elf32_Shdr;
```

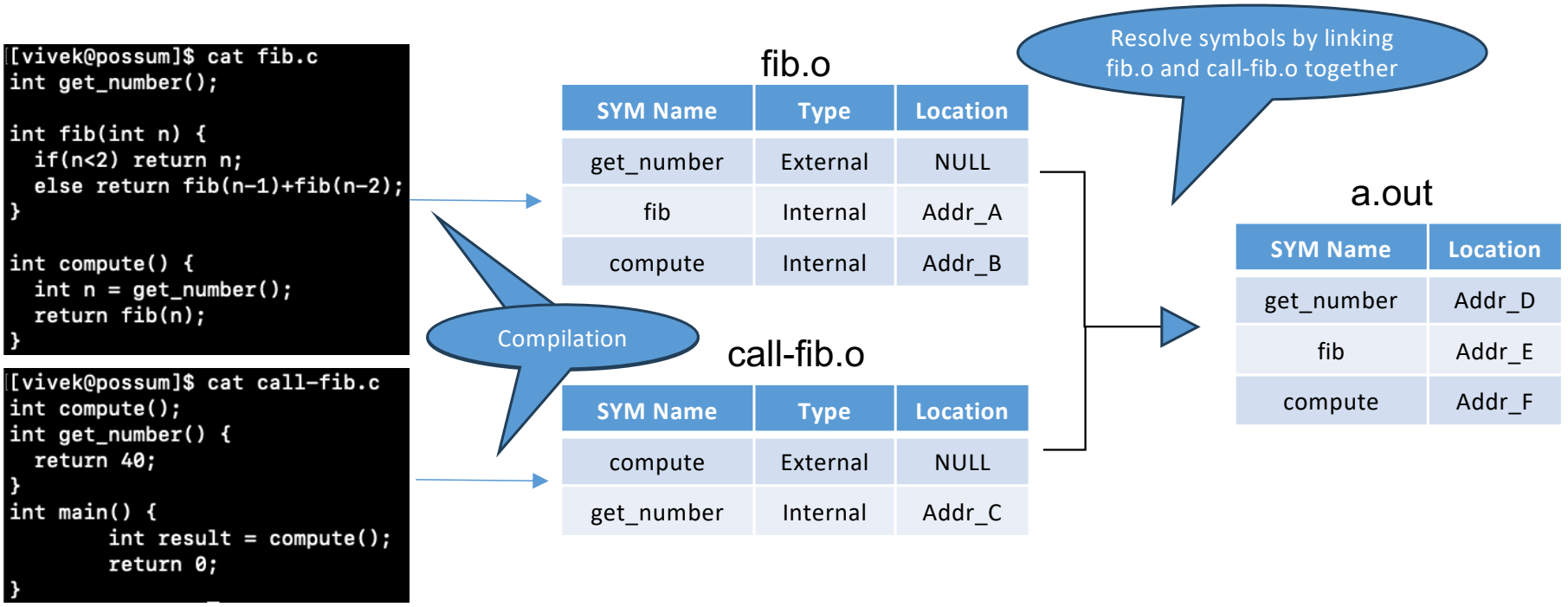
6. Iterate through all entries in the SHDR one by one
   - SHDR array starts at e_shoff
7. Find the value of sh_name in the SHDR entry
8. Name of this section will be inside the content of the section ".shstrtab and starting at the location sh_name
   - printf (%s) at that location prints the name

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
  - https://man7.org/linux/man-pages/man5/elf.5.html
  - https://www.sco.com/developers/devspecs/gabi41.pdf
  - https://wiki.osdev.org/ELF_Tutorial