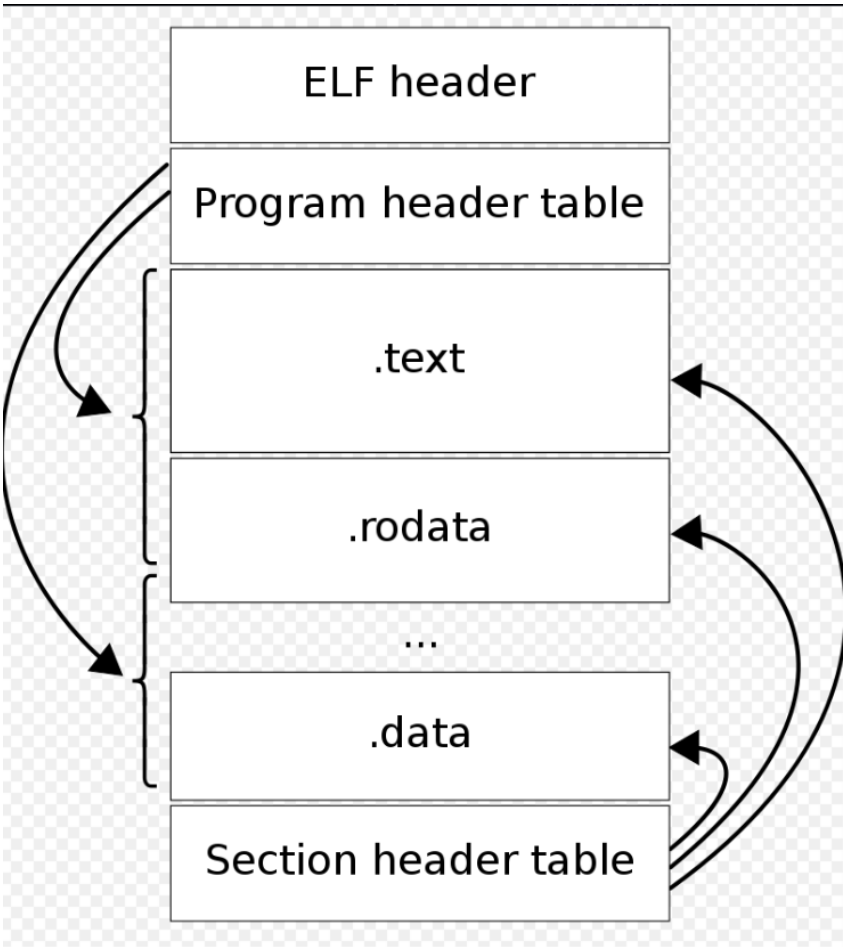
ELF

1. ELF là gì?

Elf viết tắt của Executable and Link Format – định dạng có thể thực thi và có thể liên kết

Cấu trúc của ELF



2. File header

Đây là phần đầu của file ELF có độ dài 52 byte (32bit) và 64 byte(64bit)

Figure 4-3: ELF Header

```
#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 {
        unsigned char   e_ident[EI_NIDENT];
        Elf64_Half      e_type;
        Elf64_Half      e_machine;
        Elf64_Word      e_version;
        Elf64_Addr      e_entry;
        Elf64_Off       e_phoff;
        Elf64_Off       e_shoff;
        Elf64_Word      e_flags;
        Elf64_Half      e_ehsize;
        Elf64_Half      e_phentsize;
        Elf64_Half      e_phnum;
        Elf64_Half      e_shentsize;
        Elf64_Half      e_shnum;
        Elf64_Half      e_shstrndx;
} Elf64_Ehdr;
```

Ý nghĩa các trường

- e_ident đánh dấu đây là tệp đối tượng cung cấp dữ liệu để giải mã
  - e_ident[EI_MAG0] -> e_ident[EI_MAG3] :nếu là file ELF sẽ có giá trị (0x 7F 45 4C 46)
  - e_ident[EI_CLASS] : có giá trị là 1 nếu đây là file 32bit là 2 nếu 64 bit
  - e_ident[EI_DATA] : có giá trị 1 với với kiểu dữ liệu đầu nhỏ (little endianness), 2 nếu kiểu dữ liệu đầu to (big endianness)
  - e_ident[EI_VERSION]: có giá trị 1 cho phiên bản gốc và phiên bản hiện tại của ELF
  - e_ident[EI_OSABI] : xác định hệ điều hành
  - e_ident[EI_ABIVERSION]: xác định phien bản ABI
  - e_ident[EI_PAD] : đánh dấu các byte ko sử dụng trong e_ident và có giá trị là 0
- e_type xác định loại tệp đối tượng
- e_machine: xác định cấu trúc máy tính
- e_version: có giá trị 1 cho phiên bản gốc của ELF
- e_entry : cung cấp địa chỉ ảo mà hệ thống chuyển quyền điều khiển

- e_phoff : cung cấp địa chỉ bắt đầu program header
- e_shoff : cung cấp địa chỉ bắt đầu section header
- e_flags :cờ
- e_ehsize: kích thước của ELF header 64 byte với 64 bit và 52 byte với 32 bit
- e_phentsize: chứa kích thước program header
- e_phnum: chứa số lượng mục trong program header
- e_shentsize: chứa kích thước section header
- e_shnum: chứa số lượng mục trong section header
- e_shstrndx: cung cấp section chứa tên các section

3. Program Header

Program header table là 1 mảng cấu trúc, mỗi cấu trúc mô tả một phân đoạn (segment) hoặc thông tin khác mà hệ thống cần để chuẩn bị chương trình thực thi

Trong ELF 32bit và ELF 64 bit có các trường giống nhau nhưng bố cục khác nhau (ở p_glags) và kích thước nhau

```
typedef struct {
        Elf32_Word      p_type;
        Elf32_Off       p_offset;
        Elf32_Addr      p_vaddr;
        Elf32_Addr      p_paddr;
        Elf32_Word      p_filesz;
        Elf32_Word      p_memsz;
        Elf32_Word      p_flags;
        Elf32_Word      p_align;
} Elf32_Phdr;

typedef struct {
        Elf64_Word      p_type;
        Elf64_Word      p_flags;
        Elf64_Off       p_offset;
        Elf64_Addr      p_vaddr;
        Elf64_Addr      p_paddr;
        Elf64_Xword     p_filesz;
        Elf64_Xword     p_memsz;
        Elf64_Xword     p_align;
} Elf64_Phdr;
```

   - p_type : xác định loại phân đoạn
   - p_flags: xác định cờ của phân đoạn
   - p_offset: cung cấp địa chỉ phân đoạn
   - p_vaddr: địa chỉ ảo của phân đoạn trong bộ nhớ
   - p_paddr: địa chỉ thực của phân đoạn
   - p_filesz: kích thước phân đoạn tính bằng byte trong file image
   - p_memsz : kích thước phân đoạn tính bằng byte trong memory
   - p_align: xác định căn chỉnh hay không. Nếu 0 và 1 là không cần căn chỉnh.

```
Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  LOAD           0x000000 0x08048000 0x08048000 0x00094 0x00094 R   0x1000
  LOAD           0x001000 0x08049000 0x08049000 0x00142 0x00142 R E 0x1000
  LOAD           0x002000 0x0804a000 0x0804a000 0x00002 0x00044 RW  0x1000
```

4. Section header

Section header table chứa tất cả thông tin cần thiết để xác định vị trí từng phần(section) của tệp

```
typedef struct {
    uint32_t   sh_name;
    uint32_t   sh_type;
    uint32_t   sh_flags;
    Elf32_Addr sh_addr;
    Elf32_Off  sh_offset;
    uint32_t   sh_size;
    uint32_t   sh_link;
    uint32_t   sh_info;
    uint32_t   sh_addralign;
    uint32_t   sh_entsize;
} Elf32_Shdr;

typedef struct {
    uint32_t   sh_name;
    uint32_t   sh_type;
    uint64_t   sh_flags;
    Elf64_Addr sh_addr;
    Elf64_Off  sh_offset;
    uint64_t   sh_size;
    uint32_t   sh_link;
    uint32_t   sh_info;
    uint64_t   sh_addralign;
    uint64_t   sh_entsize;
} Elf64_Shdr;
```

Cấu trúc giữa ELF 32bit và 64bit là giống nhau (khác nhau ở kích thước)

- sh_name : chứa thông tin để tìn ra tên của section
- sh_type : xác định loại section
- sh_flags: xác định cờ của section
- sh_addr: địa chỉ ảo cúa setion trong memory
- sh_offset: kích thước của section trong file image
- sh_size: kích thước tính bằng byte của section trong file image

- sh_link: liên kết chỉ mục bảng tiêu đề phần
- sh_info: thông tin bổ sung của section
- sh_addralign: chứa liên kết cần thiết của section
- sh_entsize: Chứa kích thướng tính bằng byte của section. Nếu section không có kích thước cố định giá trị là 0

*cách tìm vị trí chứa tên các section

Tính header_table_entry_offset

Here is how you would get to section name string table:

- The `e_shstrndx` field of the ELF Executable Header (EHDR) specifies the index of the section header table entry describing the string table containing section names.
- The `e_shentsize` field of the EHDR specifies the size in bytes of each section header table entry.
- The `e_shoff` field of the EHDR specifies the file offset to the start of the section header table.

Thus the header entry for the string table will be at file offset:

```
header_table_entry_offset = (e_shentsize * e_shstrndx) + e_shoff
```

Offset_name_section = sh_name + header_table_entry_offset