ELF 文件格式

1. 了解 ELF 文件格式有什么用
2. 如何加载一个 ELF 文件
3. 如何构建一个不依赖 C 标准库的 ELF

```
/*
 * Magic values required to use _reboot() system call.
 */

#define LINUX_REBOOT_MAGIC1     0xfee1dead
#define LINUX_REBOOT_MAGIC2     672274793
#define LINUX_REBOOT_MAGIC2A    85072278
#define LINUX_REBOOT_MAGIC2B    369367448
#define LINUX_REBOOT_MAGIC2C    537993216
```

了解 ELF 文件格式有什么用？

1. 修改可执行文件
2. 为学习编译器、操作系统、虚拟机等打下基础
3. 实现自己的可执行文件格式

COFF

共同对象文件格式（英语：Common Object File Format，缩写为COFF），又称**通用目标文件格式**，是一种用于可执行文件、目标代码、共享库（shared library）的文件格式，使用于类UNIX系统上。它最早使用于UNIX System V上，用来取代先前的a.out格式，后来又发展出XCOFF与ECOFF。

在多数类UNIX系统上，这个格式已被ELF格式所取代。某些类Unix系统，微软公司的Windows系统（PE 格式），可扩展固件接口（EFI）以及某些嵌入式系统中仍在使用COFF文件格式或它的变种。

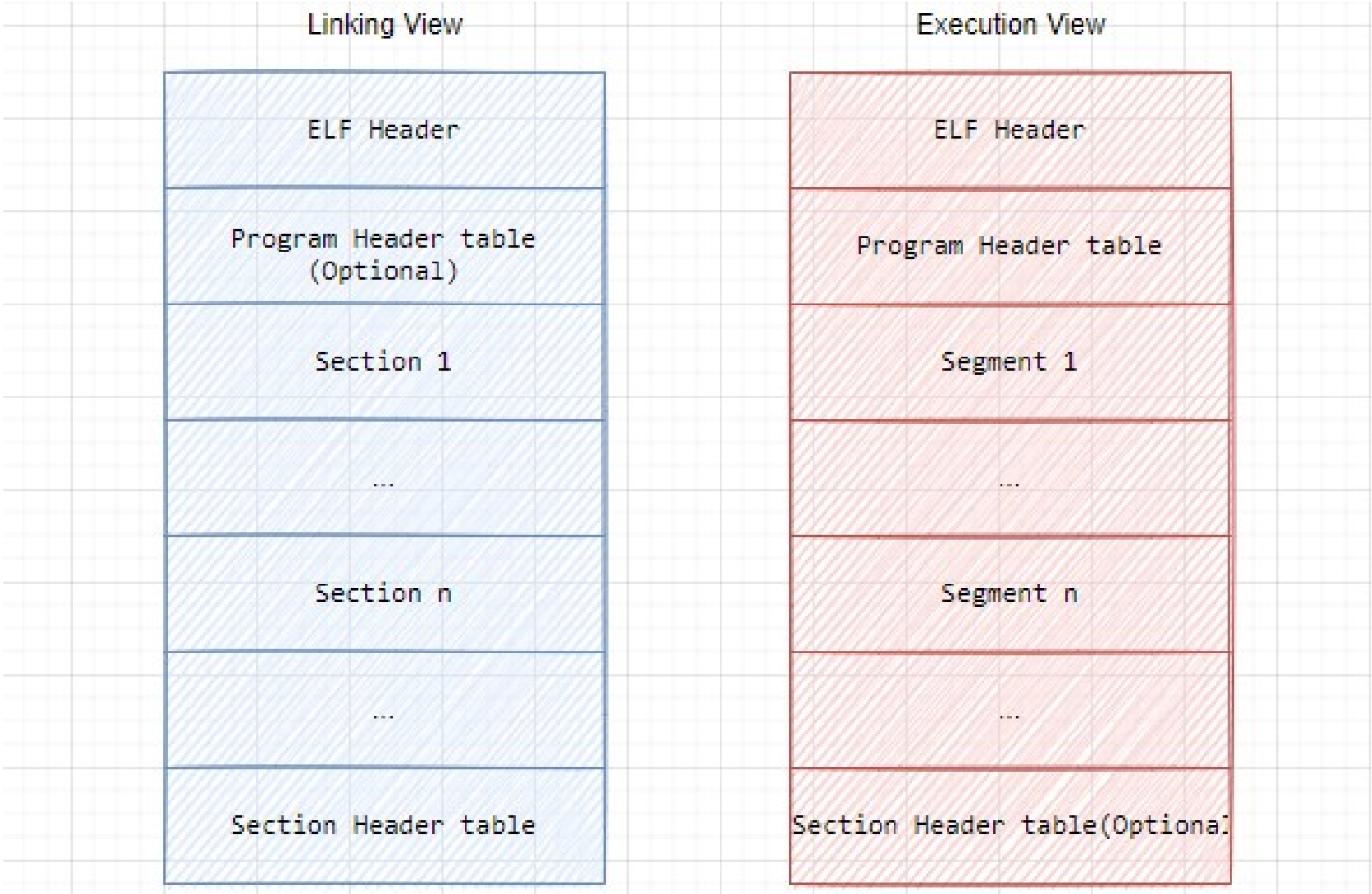
COFF

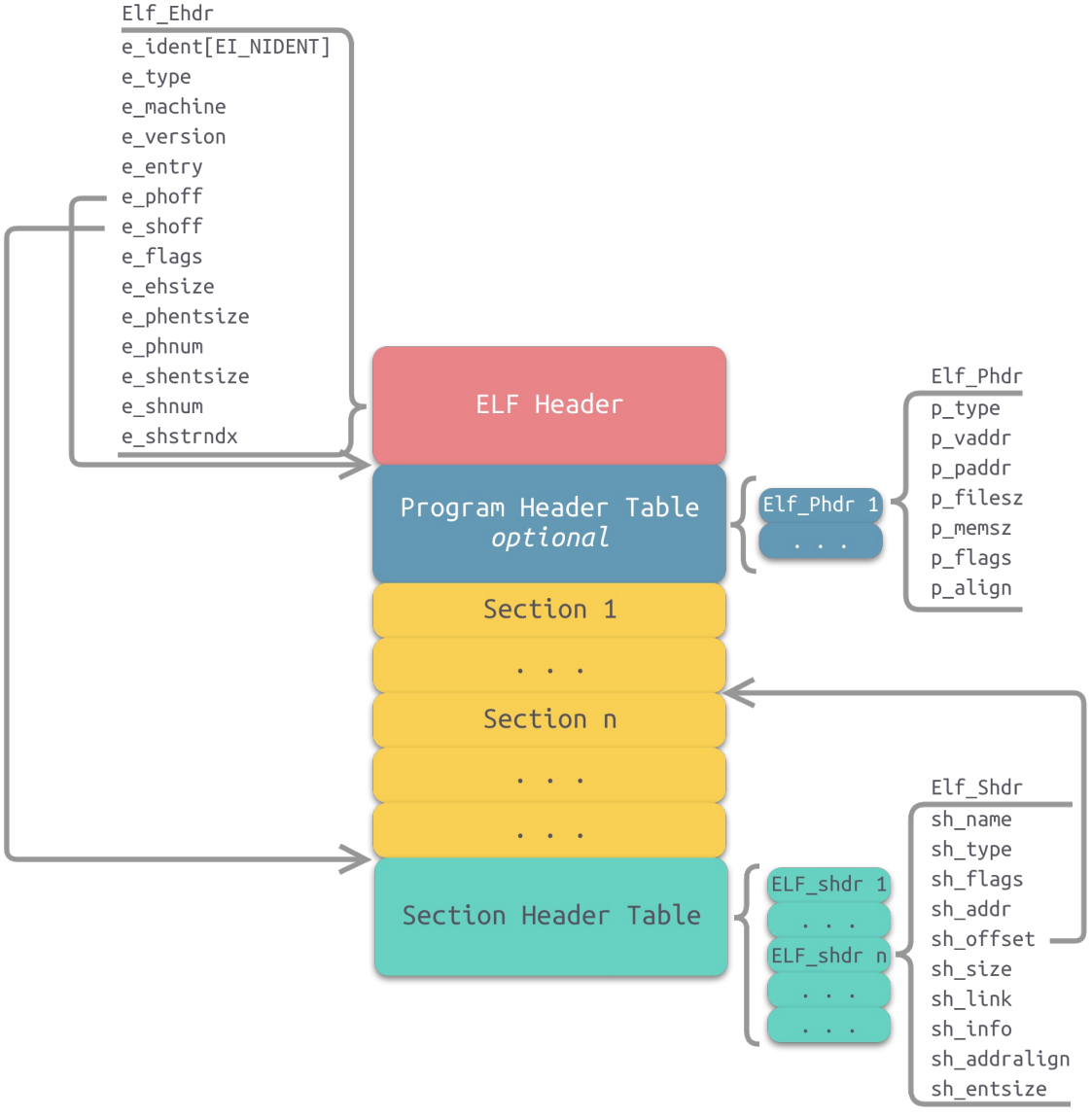
扩展名	无、.o、.obj
互联网媒体类型	application/x-coff、application/x-coffexec
开发者	美国电话电报公司
格式类型	二进制可执行文件、目标代码、函数库
扩展为	XCOFF、ECOFF、可移植可执行

这是一篇与电脑存储设备相关的小作品。您可以通过编辑或修订扩充其内容。

查 • 论 • 编 **可执行文件和目标文件格式** [折叠]

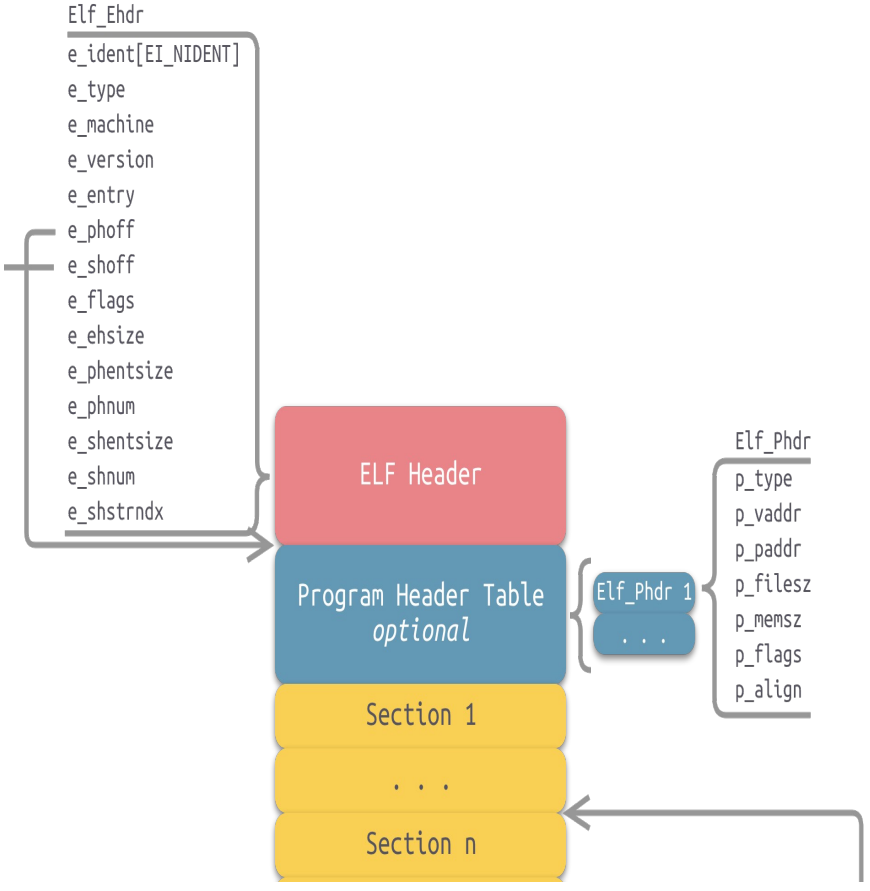
a.out • AIF • **COFF** • CMD • COM • ECOFF • ELF • GOFF • Hunk • Mach-O • MZ • NE • OMF • OS/360 • PE • PEF • XCOFF


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 n
...
Section Header table(Optiona


Elf_Ehdr
e_ident[EI_NIDENT]
e_type
e_machine
e_version
e_entry
e_phoff
e_shoff
e_flags
e_ehsize
e_phentsize
e_phnum
e_shentsize
e_shnum
e_shstrndx
ELF Header
Program Header Table
optional
Elf_Phdr 1
. . .
Elf_Phdr
p_type
p_vaddr
p_paddr
p_filesz
p_memsz
p_flags
p_align
Section 1
. . .
Section n
. . .
. . .
Section Header Table
ELF_shdr 1
. . .
ELF_shdr n
. . .
. . .
Elf_Shdr
sh_name
sh_type
sh_flags
sh_addr
sh_offset
sh_size
sh_link
sh_info
sh_addralign
sh_entsize

字段	解释	典型值
e_ident	魔数，用于检查是否是 ELF 文件	0x7f, ELF
e_type	文件类型	ET_EXEC
e_machine	设备类型	EM_AMD64
e_entry	入口地址，从这个地址开始执行指令	_start
e_phoff	Program Headers 在文件中的偏移	
e_phnum	Program Headers 的数量	
e_shoff		
e_shnum		
e_shstrndx	Section Header 字符串表在 Section Header Table 中的索引	

```c
#define Ehdr Elf64_Ehdr
#define Phdr Elf64_Phdr
#define ELFCLASS ELFCLASS64

uint8_t elf_magic[] = {0x7f,'E','L','F'};
uintptr_t elf_loader(uint8_t *elf_base) {
    assert(elf_base != NULL);

    /* parse header */
    Ehdr *header = (Ehdr*)elf_base;
    assert(memcmp(header->e_ident, elf_magic, sizeof(elf_magic)) == 0);
    assert(header->e_machine == EM_RISCV);
    assert(header->e_ident[EI_CLASS] == ELFCLASS);
    assert(header->e_type == ET_EXEC || header->e_type == ET_DYN);

    /* find segment */
    Phdr *segments = (Phdr*)(elf_base + header->e_phoff);
    size_t phnum = header->e_phnum, ldnum = 0;
    Log("find %p segment(s) from elf", phnum);
    assert(phnum);

    /* load segment to memory */
    for (int i=0; i < phnum; i++) {
        Phdr *segment = &segments[i];
        // ignore p_type etc.
        uint8_t *dst = (uint8_t*)segment->p_vaddr,
                *src = elf_base + segment->p_offset;
        size_t len = segment->p_filesz,
               remain = segment->p_memsz - len;

        if (segment->p_type == PT_LOAD) {
            panic_on(!dst, "wrong vaddr");
            panic_on(segment->p_filesz > segment->p_memsz, "wrong filesz");
            assert(len);

            memcpy(dst, src, len);
            memset(dst+len, '\0', remain);
            ldnum++;
        }
    }

    Log("load %p segment(s)", ldnum);
    return header->e_entry;
}
```




你得重定位后才能调试和
使用库函数
我需要调试才能写出重定位
的代码
那就去调试啊
重定位完成后才能调试打
断点输出变量
那就去重定位啊


glibc
功能丰富
兼容性强
安全性好
mylibc
自己写的
想改就改

```
void printf (const char *format, ...) {
  char **arg = (char **) &format;
  int c;
  char buf[20];
  arg++;

  while ((c = *format++) != 0) {
      if (c != '%')
        putchar (c);
      else {
          char *p, *p2;
          int pad0 = 0, pad = 0;

          c = *format++;
          if (c == '0') {
              pad0 = 1;
              c = *format++;
            }

          if (c >= '0' && c <= '9') {
              pad = c - '0';
              c = *format++;
            }
```

```
          switch (c) {
            case 'd':
            case 'u':
            case 'x':
              itoa (buf, c, *((int *) arg++));
              p = buf;
              goto string;
              break;

            case 's':
              p = *arg++;
              if (! p)
                p = "(null)";

            string:
              for (p2 = p; *p2; p2++);
              for (; p2 < p + pad; p2++)
                putchar (pad0 ? '0' : ' ');
              while (*p)
                putchar (*p++);
              break;

            default:
              putchar (*((int *) arg++));
              break;
            }
        }
    }
```

https://www.gnu.org/software/grub/manual/multiboot/multiboot.html