


# ELF可执行目标文件

南京大学
计算机科学与技术系
袁春风
email: cfyuan@nju.edu.cn
2015.6

# 回顾：可重定位目标文件格式

ELF 头

- ✓ 包括16字节标识信息、文件类型 (.o, exec, .so)、机器类型（如 IA-32）、节头表的偏移、节头表的表项大小以及表项个数

.text 节

- ✓ 编译后的代码部分

.rodata 节

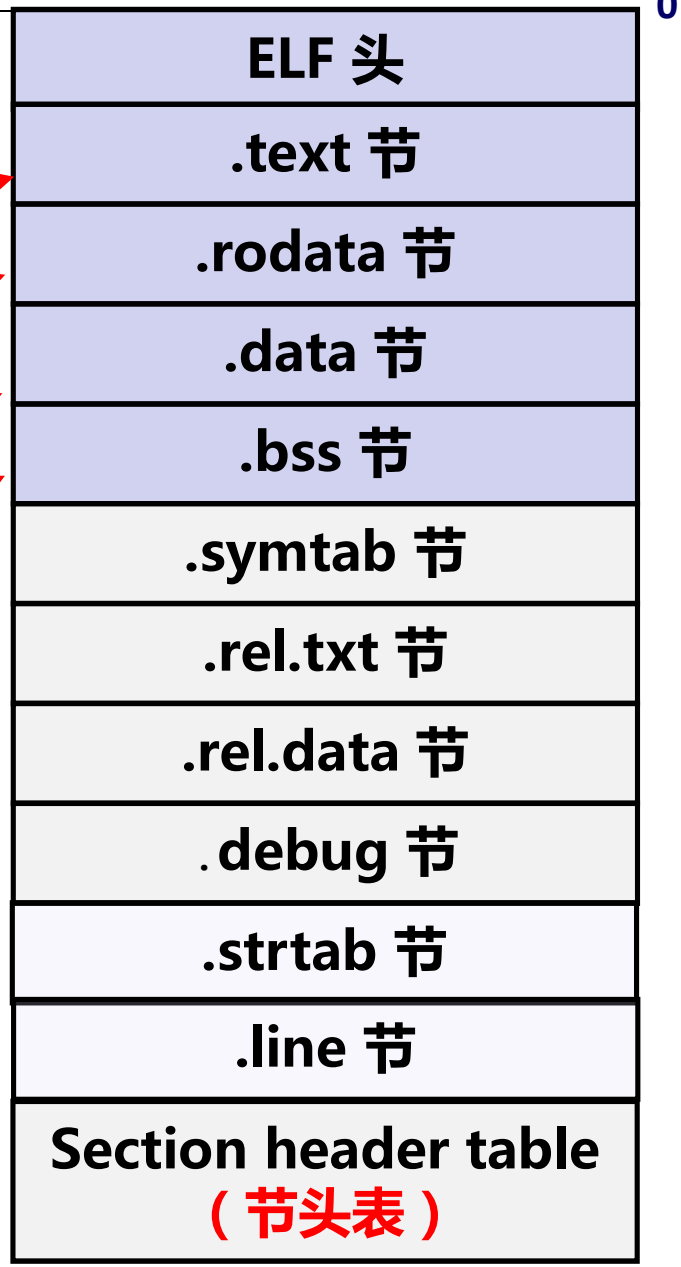
- ✓ 只读数据，如 printf 格式串、switch 跳转表等

.data 节

- ✓ 已初始化的全局变量

.bss 节

- ✓ 未初始化全局变量，仅是占位符，不占据任何实际磁盘空间。区分初始化和非初始化是为了空间效率



# 回顾：可重定位目标文件格式

.symtab 节
- ✓ 存放函数和全局变量（符号表）信息，它不包括局部变量

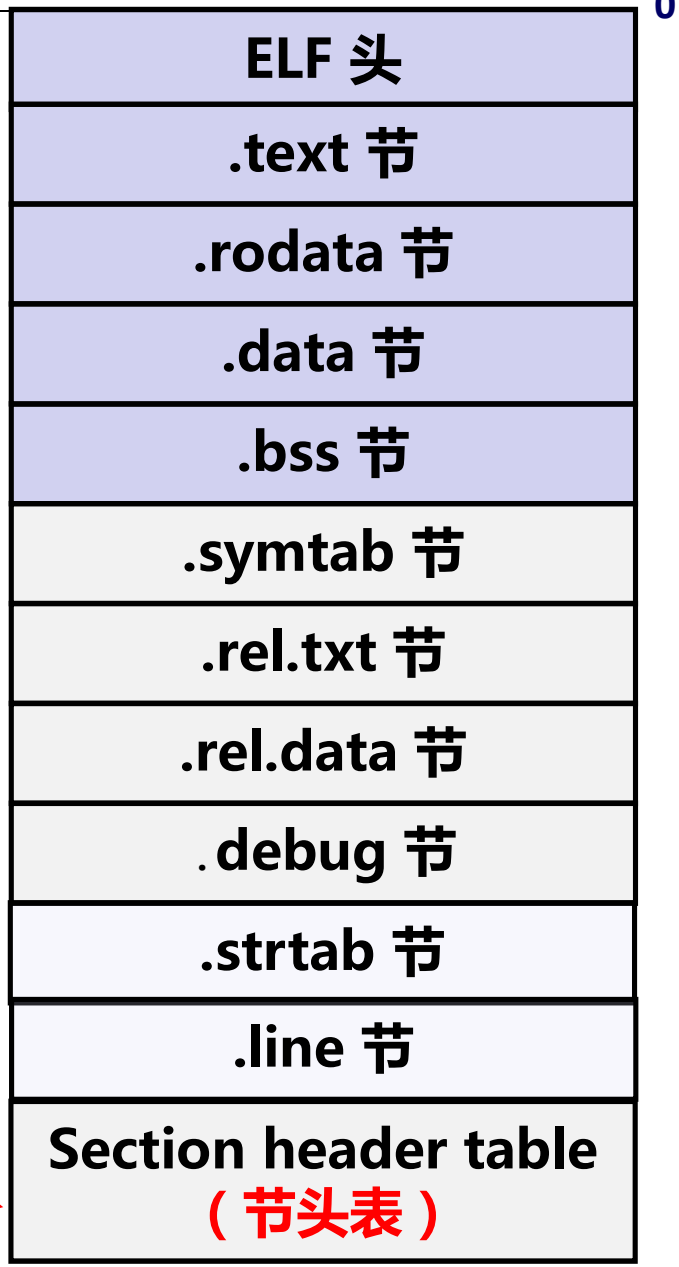
.rel.text 节
- ✓ .text节的重定位信息，用于重新修改代码段的指令中的地址信息

.rel.data 节
- ✓ .data节的重定位信息，用于对被模块使用或定义的全局变量进行重定位的信息

.debug 节
- ✓ 调试用符号表 (gcc -g)

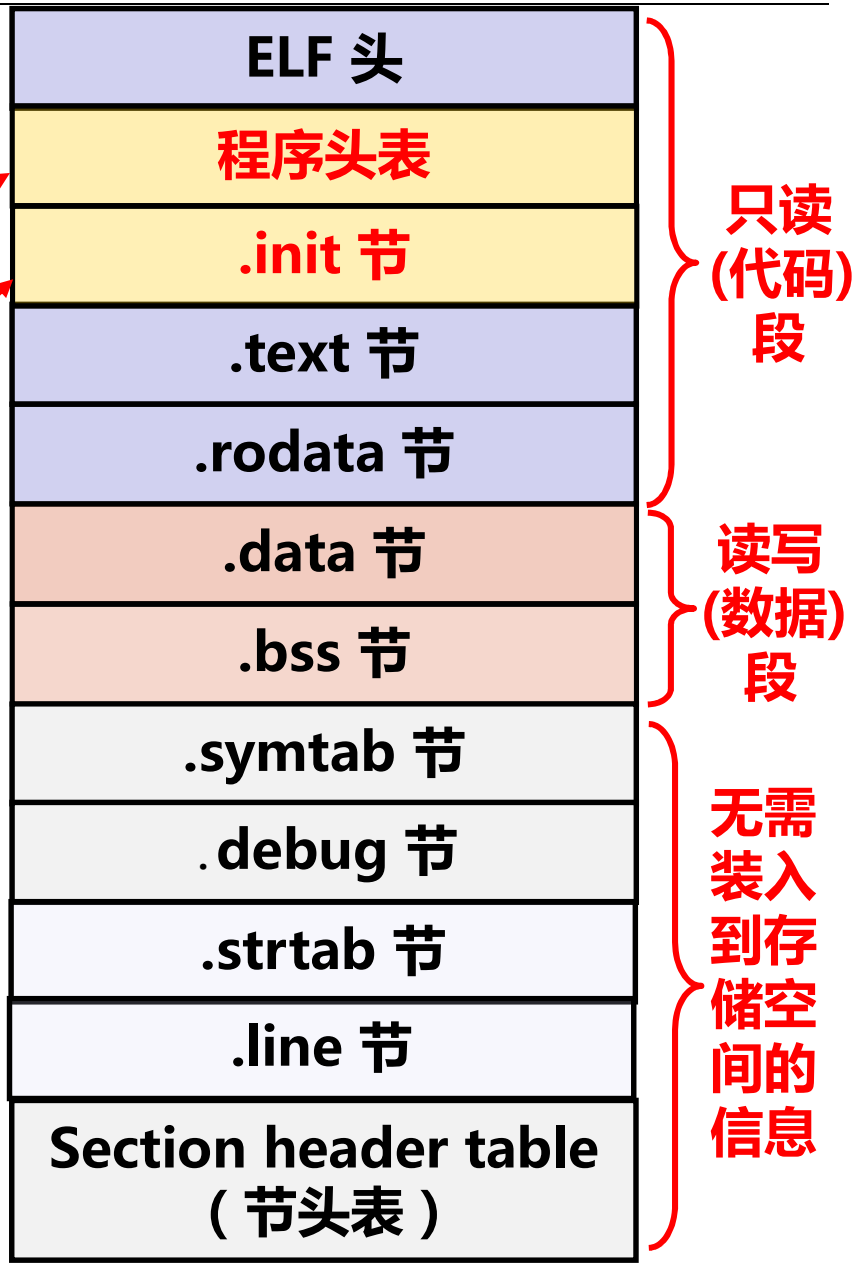
strtab 节
- ✓ 包含symtab和debug节中符号及节名

Section header table（节头表）
- ✓ 每个节的节名、偏移和大小



# 可执行目标文件格式

- 与可重定位文件稍有不同：
  - ELF头中字段e_entry给出执行程序时第一条指令的地址，而在可重定位文件中，此字段为0
  - 多一个程序头表，也称段头表（segment header table），是一个结构数组
  - 多一个.init节，用于定义_init函数，该函数用来进行可执行目标文件开始执行时的初始化工作
  - 少两个.rel节（无需重定位）



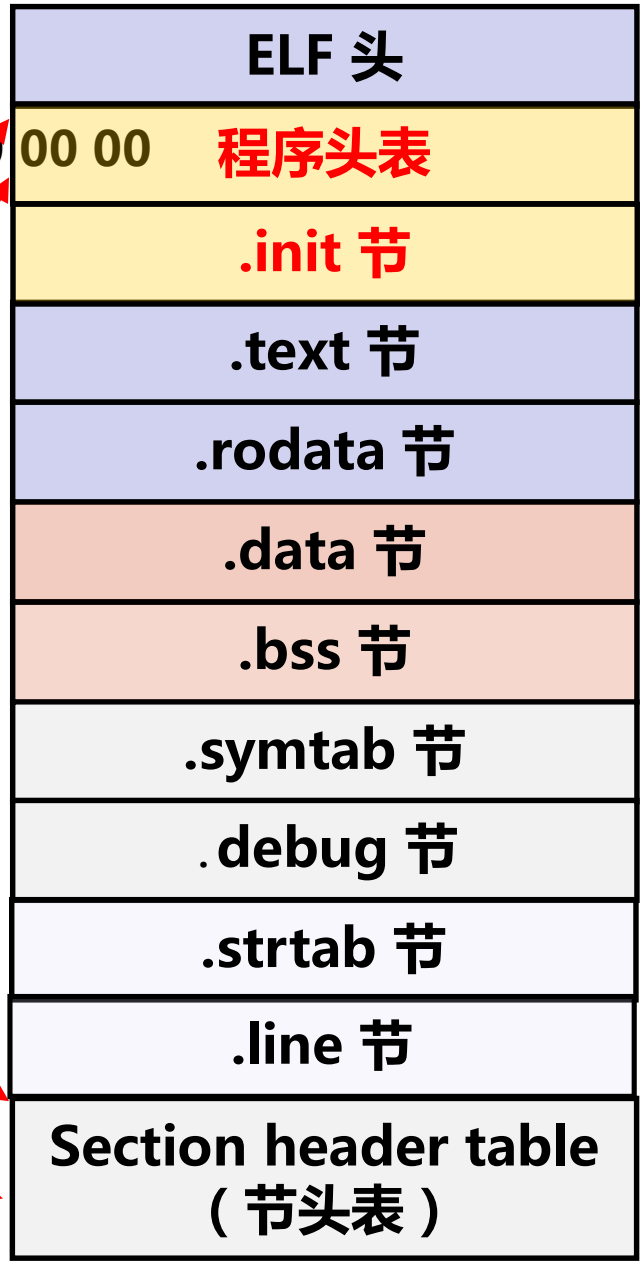
# ELF头信息举例

**$ readelf -h main** 可执行目标文件的ELF头

```
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               x8048580
  Start of program headers:          52 (bytes into file)
  Start of section headers:          3232 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         8
  Size of section headers:           40 (bytes)
  Number of section headers:         29
  Section header string table index: 26
```

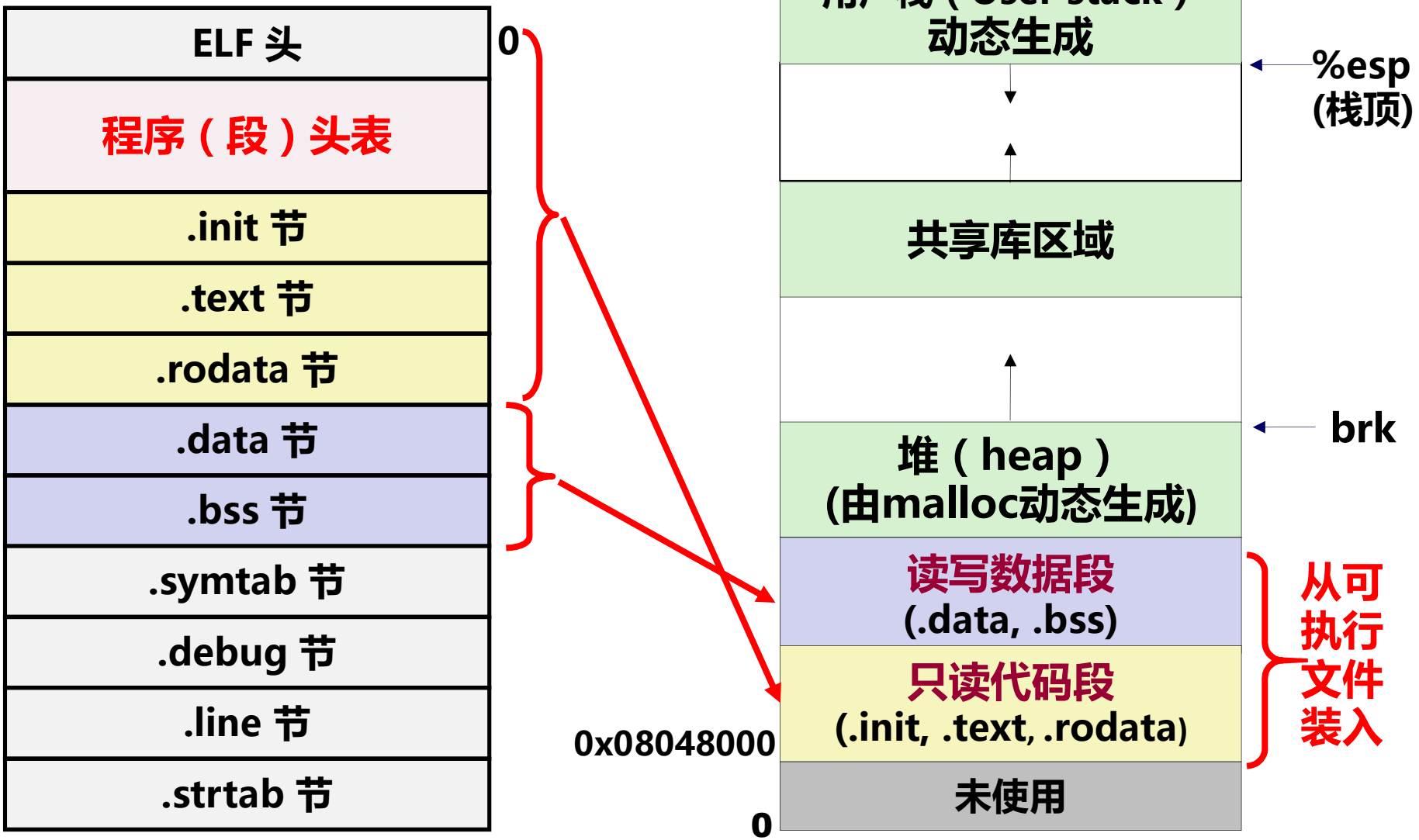
8x32B

29x40B



# 可执行文件的存储器映像

程序(段)头表描述如何映射



# 可执行文件中的程序头表

```
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;
```

程序头表描述可执行文件中的节与虚拟空间中的存储段之间的映射关系

一个表项（32B）说明虚拟地址空间中一个连续的段或一个特殊的节

以下是某可执行目标文件程序头表信息

有8个表项，其中两个为可装入段（即Type=LOAD）

$ readelf –l main

```
Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR           0x000034 0x08048034 0x08048034 0x00100 0x00100 R E 0x4
  INTERP         0x000134 0x08048134 0x08048134 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
  LOAD           0x000000 0x08048000 0x08048000 0x004d4 0x004d4 R E 0x1000
  LOAD           0x000f0c 0x08049f0c 0x08049f0c 0x00108 0x00110 RW  0x1000
  DYNAMIC        0x000f20 0x08049f20 0x08049f20 0x000d0 0x000d0 RW  0x4
  NOTE           0x000148 0x08048148 0x08048148 0x00044 0x00044 R   0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RW  0x4
  GNU_RELRO      0x000f0c 0x08049f0c 0x08049f0c 0x000f4 0x000f4 R   0x1
```

# 可执行文件中的程序头表

```
Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR           0x000034 0x08048034 0x08048034 0x00100 0x00100 R E 0x4
  INTERP         0x000134 0x08048134 0x08048134 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
  LOAD           0x000000 0x08048000 0x08048000 0x004d4 0x004d4 R E 0x1000
  LOAD           0x000f0c 0x08049f0c 0x08049f0c 0x00108 0x00110 RW  0x1000
  DYNAMIC        0x000f20 0x08049f20 0x08049f20 0x000d0 0x000d0 RW  0x4
  NOTE           0x000148 0x08048148 0x08048148 0x00044 0x00044 R   0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RW  0x4
  GNU_RELRO      0x000f0c 0x08049f0c 0x08049f0c 0x000f4 0x000f4 R   0x1
```

SKIP

**第一可装入段：第0x00000~0x004d3字节（包括ELF头、程序头表、.init、.text和.rodata节），映射到虚拟地址0x8048000开始长度为0x4d4字节的区域，按0x1000=$2^{12}$=4KB对齐，具有只读/执行权限（Flg=RE），是只读代码段。**

**第二可装入段：第0x000f0c开始长度为0x108字节的.data节，映射到虚拟地址0x8049f0c开始长度为0x110字节的存储区域，在0x110=272B存储区中，前0x108=264B用.data节内容初始化，后面272-264=8B对应.bss节，初始化为0，按0x1000=4KB对齐，具有可读可写权限（Flg=RW），是可读写数据段。**

# 可执行文件的存储器映像