Lesson 7
Linking 1

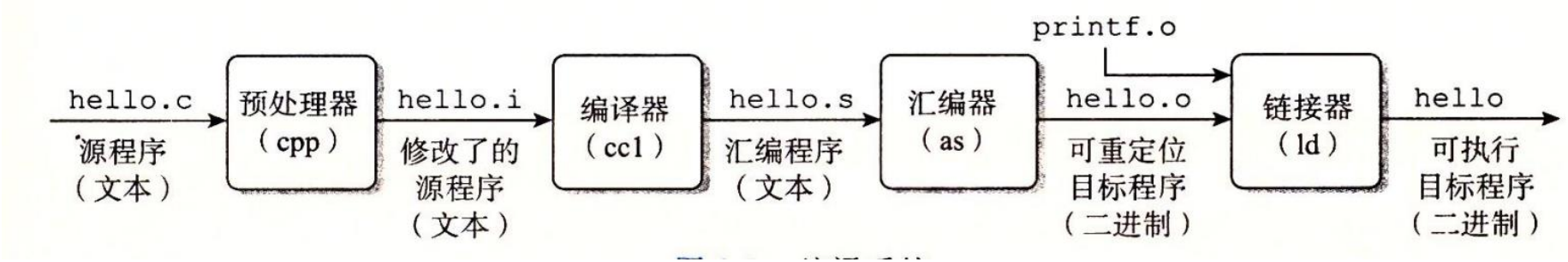
ICS Seminar #9

张龄心

Nov 15, 2023

目标文件

- 回顾: .c –> (cpp) -> .i -> (cc1) -> .s -> (as) -> .o -> (ld) -> .exe



- 静态链接: 符号解析 + 重定位
- 目标文件 Object Files
  - 可重定位目标文件 Relocatable: .o
  - 可执行目标文件 Executable: .out
  - 共享目标文件 Shared: .so file

目标文件

- ELF可重定位目标文件
  - 需要记忆每个节的具体内容
  - 区别: ELF可执行目标文件
- .data和.bss的区别?
  - .data: 已初始化的global/static
  - .bss: 未初始化/初始化为0的global/static

    - 仅仅是占位符, 不占实际空间
- COMMON和.bss的区别?
  - COMMON: 未初始化的global
  - .bss: 未初始化的static, 初始化为0的global/static

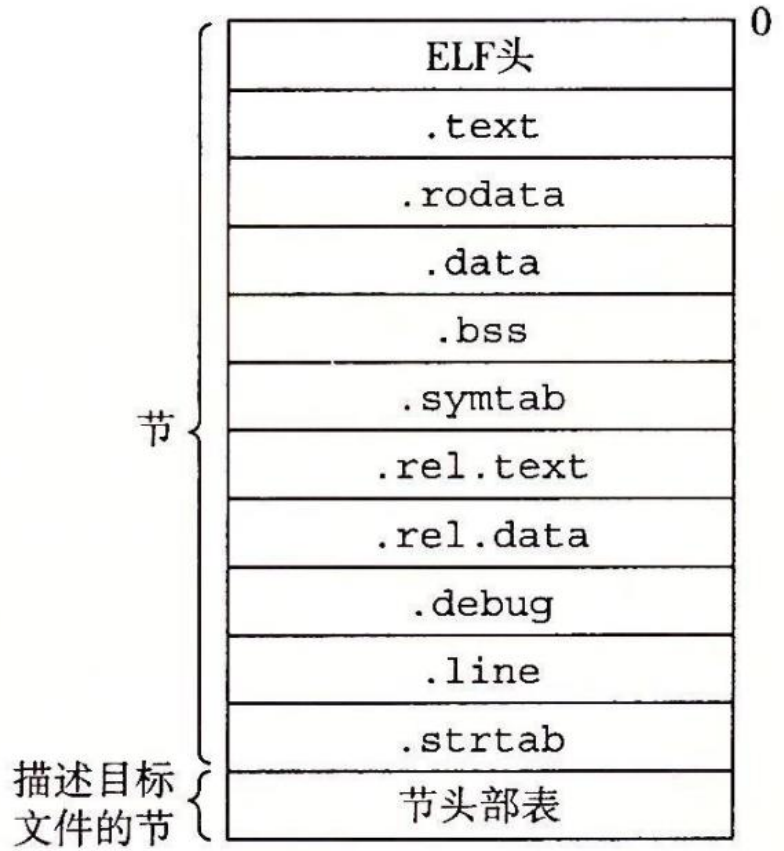


图 7-3 典型的 ELF 可重定位目标文件

符号表和符号

- 符号表 .symtab里的符号:

  全局符号 / 外部符号 / 局部符号 (global / external / local)
- 全局符号: 模块m定义的, 所有人都能用的符号
  - 非static/静态的函数和全局变量
- 外部符号: 其他模块定义的全局符号
- 局部符号 local symbols: 模块m定义, 且只能自己用的符号
  - 带static的函数和全局变量
- 符号表不包含”本地变量”(比如函数中定义的变量)

  - 因为这些变量用的时候暂存在寄存器/栈上就行, 没必要储存在符号表里

  - 后面进行”符号解析”时需要查阅的符号, 才会被储存在符号表的符号

符号解析

- 全局符号分为: 强符号 / 弱符号
  - 强符号: 函数; 已初始化的global变量
  - 弱符号: 未初始化的global变量
- 规则: 强符号彼此不能重名; 强符号>弱符号; 多个弱符号重名则任选一个

重定位

- 重定位条目
  - R_X86_64_PC32: 重定位一个使用32位PC相对地址的引用
  - R_X86_64_32: 重定位一个使用32位绝对地址的引用
- x86-64小型代码模型
  - 代码和数据总大小<2GB, 故32位相对地址足以表示

重定位

- 重定位算法

```
1   foreach section s {
2       foreach relocation entry r {
3           refptr = s + r.offset;  /* ptr to reference to be relocated */
4
5           /* Relocate a PC-relative reference */
6           if (r.type == R_X86_64_PC32) {
7               refaddr = ADDR(s) + r.offset; /* ref's run-time address */
8               *refptr = (unsigned) (ADDR(r.symbol) + r.addend - refaddr);
9           }
10
11          /* Relocate an absolute reference */
12          if (r.type == R_X86_64_32)
13              *refptr = (unsigned) (ADDR(r.symbol) + r.addend);
14      }
15  }
```

目标文件

- ELF可执行目标文件

- 与ELF目标文件的区别
  - 多了段头部表和.init节
  - 没有 .rel.text 和 .rel.data
  - 没有.debug节

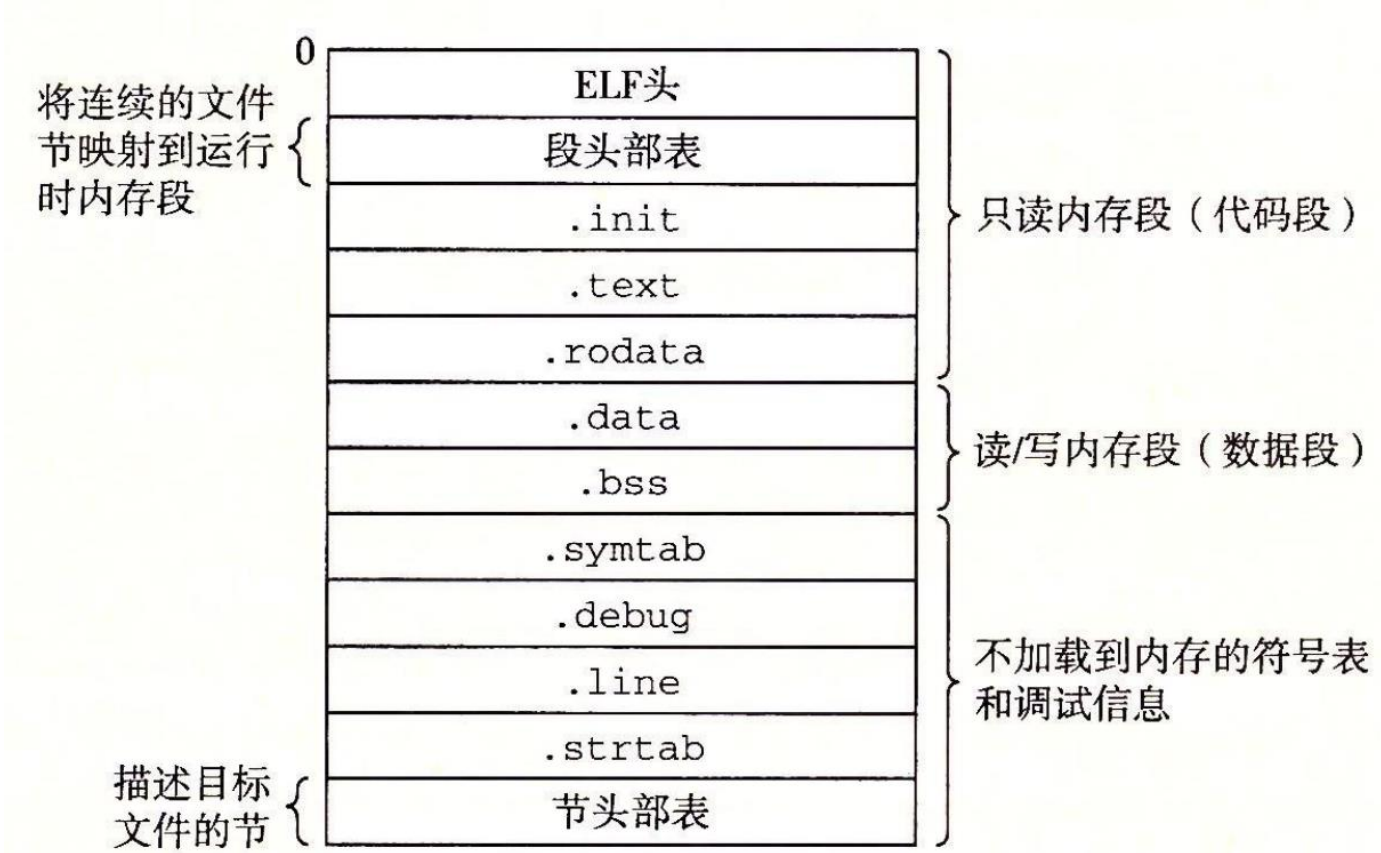


图 7-13　典型的 ELF 可执行目标文件

Thank you!