0x4004e8。那么 callq 指令中对 swap 的重定位引用的值是什么？

7.8 可执行目标文件

我们已经看到链接器如何将多个目标文件合并成一个可执行目标文件。我们的示例 C 程序，开始时是一组 ASCII 文本文件，现在已经被转化为一个二进制文件，且这个二进制文件包含加载程序到内存并运行它所需的所有信息。图 7-13 概括了一个典型的 ELF 可执行文件中的各类信息。

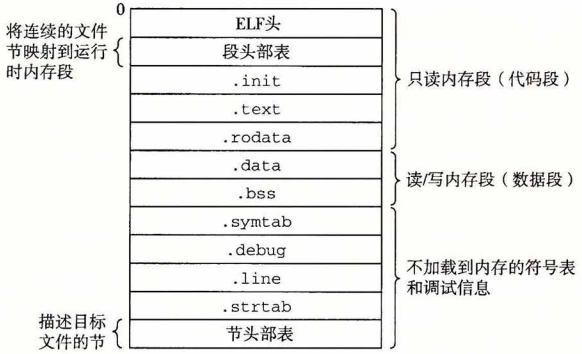

图 7-13 典型的 ELF 可执行目标文件

可执行目标文件的格式类似于可重定位目标文件的格式。ELF 头描述文件的总体格式。它还包括程序的入口点(entry point)，也就是当程序运行时要执行的第一条指令的地址。.text、.rodata 和 .data 节与可重定位目标文件中的节是相似的，除了这些节已经被重定位到它们最终的运行时内存地址以外。.init 节定义了一个小函数，叫做 _init，程序的初始化代码会调用它。因为可执行文件是完全链接的(已被重定位)，所以它不再需要 .rel 节。

ELF 可执行文件被设计得很容易加载到内存，可执行文件的连续的片(chunk)被映射到连续的内存段。程序头部表(program header table)描述了这种映射关系。图 7-14 展示了可执行文件 prog 的程序头部表，是由 OBJDUMP 显示的。

code/link/prog-exe.d

```
 Read-only code segment
1 LOAD off    0x0000000000000000 vaddr 0x0000000000400000 paddr 0x0000000000400000 align 2**21
2     filesz 0x000000000000069c memsz 0x000000000000069c flags r-x

 Read/write data segment
3 LOAD off    0x00000000000000df8 vaddr 0x0000000000600df8 paddr 0x0000000000600df8 align 2**21
4     filesz 0x0000000000000228 memsz 0x0000000000000230 flags rw-
```

code/link/prog-exe.d

图 7-14 示例可执行文件 prog 的程序头部表

off：目标文件中的偏移；vaddr/paddr：内存地址；align：对齐要求；filesz：目标文件中的段大小；memsz：内存中的段大小；flags：运行时访问权限。

从程序头部表，我们会看到根据可执行目标文件的内容初始化两个内存段。第 1 行和