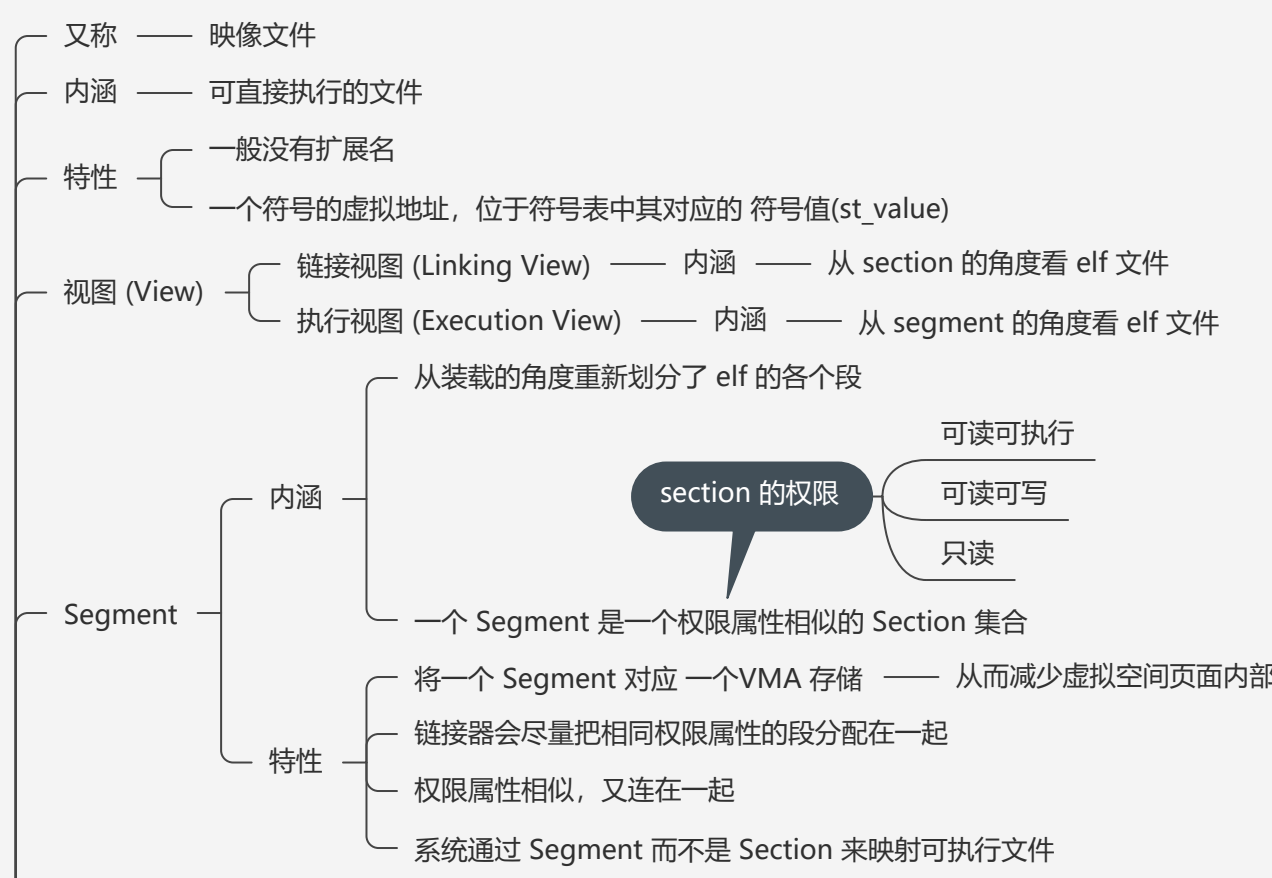


可执行文件(Executable File)



内涵 —— 描述 segment  
描述 elf 文件该如何被操作系统映射到进程的虚拟空间

```
typedef struct
{
  Elf32_Word  p_type;          /* Segment type */
  Elf32_Off  p_offset;        /* Segment file offset */
  Elf32_Addr  p_vaddr;        /* Segment virtual address */
  Elf32_Addr  p_paddr;        /* Segment physical address */
  Elf32_Word  p_filesz;       /* Segment size in file */
  Elf32_Word  p_memsz;        /* Segment size in memory */
  Elf32_Word  p_flags;        /* Segment flags */
  Elf32_Word  p_align;        /* Segment alignment */
} Elf32_Phdr;
```

成员	含义
p_type	"Segment" 的类型，基本上我们在这里只关注 "LOAD" 类型的 "Segment"。"LOAD" 类型的常量为 1。还有几个类型诸如 "DYNAMIC"、"INTERP" 等我们在介绍 ELF 动态链接时还会碰到
p_offset	"Segment" 在文件中的偏移
p_vaddr	"Segment" 的第一个字节在进程虚拟地址空间的起始位置。整个程序头表中，所有 "LOAD" 类型的元素按照 p_vaddr 从小到大排列
p_paddr	"Segment" 的物理装载地址。我们在本书的第 2 部分已经碰到过一个叫做 LMA (Load Memory Address) 的概念，这个物理装载地址就是 LMA。p_paddr 的值在一般情况下跟 p_vaddr 是一样的
p_filesz	"Segment" 在 ELF 文件中所占空间的长度。它的值可能是 0，因为有可能这个 "Segment" 在 ELF 文件中不存在内容
p_memsz	"Segment" 在进程虚拟地址空间中所占用的长度。它的值也可能是 0
p_flags	"Segment" 的权限属性，比如可读 "R"，可写 "W" 和可执行 "X"
p_align	"Segment" 的对齐属性。实际对齐字节等于 2 的 p_align 次。比如 p_align 等于 10，那么实际的对齐属性就是 2 的 10 次方，即 1024 字节

结构 —— Elf32\_Phdr 的数组

程序头 (Program Header)

命令 —— readelf -l <file>

```
lnhann@ubun:~/Repo_Proj/Coding/cpp-untitled/dynamic$ readelf -l pi

Elf file type is DYN (Shared object file)
Entry point 0x1090
There are 12 program headers, starting at offset 52

Program Headers:
Type           Offset  VirtAddr  PhysAddr  FileSiz MemSiz  Flg Align
PHDR           0x000034 0x00000034 0x00000034 0x00180 0x00180 R  0x4
INTERP        0x0001b4 0x000001b4 0x000001b4 0x00013 0x00013 R  0x1
[Requesting program interpreter: /lib/ld-linux.so.2]
LOAD          0x000000 0x00000000 0x00000000 0x00400 0x00400 R  0x1000
LOAD          0x001000 0x00001000 0x00001000 0x002a4 0x002a4 R E 0x1000
LOAD          0x002000 0x00002000 0x00002000 0x00194 0x00194 R  0x1000
LOAD          0x002ed0 0x00003ed0 0x00003ed0 0x00138 0x0013c RW 0x1000
DYNAMIC       0x002ed8 0x00003ed8 0x00003ed8 0x00100 0x00100 RW 0x4
NOTE         0x0001c8 0x000001c8 0x000001c8 0x00060 0x00060 R  0x4
GNU_PROPERTY 0x0001ec 0x000001ec 0x000001ec 0x0001c 0x0001c R  0x4
GNU_EH_FRAME 0x002008 0x00002008 0x00002008 0x00054 0x00054 R  0x4
GNU_STACK    0x000000 0x00000000 0x00000000 0x00000 0x00000 RW 0x10
GNU_RELRO    0x002ed0 0x00003ed0 0x00003ed0 0x00130 0x00130 R  0x1

Section to Segment mapping:
Segment Sections...
00
01 .interp
02 .interp .note.gnu.build-id .note.gnu.property .note.ABI-tag .gnu.hash .dynsym .dynstr .gnu.version .gnu.version_r .rel.dyn .rel.plt
03 .init .plt .plt.got .plt.sec .text .fini
04 .rodata .eh_frame_hdr .eh_frame
05 .init_array .fini_array .dynamic .got .data .bss
06 .dynamic
07 .note.gnu.build-id .note.gnu.property .note.ABI-tag
08 .note.gnu.property
09 .eh_frame_hdr
10
11 .init_array .fini_array .dynamic .got
```

查看

