《openEuler内核编程》

课程讲稿

第二章 第2讲

操作系统基本结构和映像布局

软件所制

第二章 第2讲 操作系统基本结构和映像布局

**学时：**1学时

**教学目的：**学习ELF格式的组成部分。了解linux映像文件的组成，学会使用readelf、objdump命令来查看文件的elf头信息及符号表。

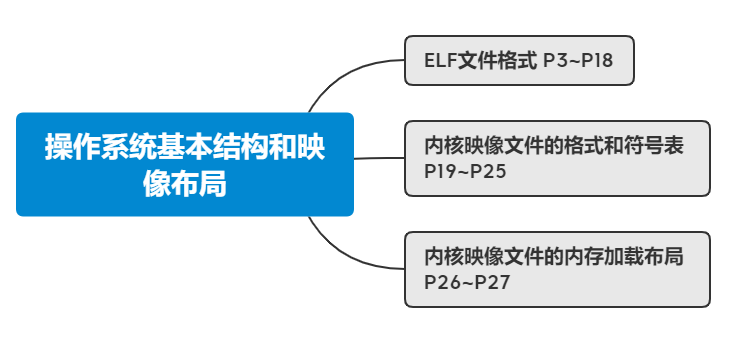
**课程时间线：**



**课外参考读物：**

1. <http://chuquan.me/2018/05/21/elf-introduce/>

**知识框图：**



**PPT讲稿：**

3. 在Linux操作系统中，通过源代码编译完成的二进制文件，有一个统一的格式，叫做ELF文件。全称是Executable and Linkable Format。翻译成中文就是可执行和可链接格式。这一节我们讲内核映像布局，内核映像也是二进制文件，所以要先学习Linux系统下二进制文件的格式。
4. 好，同样是ELF文件，一共有四种类型。  
   第一种是可执行文件，Linux下的可执行文件一般都没有扩展名，对应Windows系统下就是.exe文件。
5. 第二种是可重定位文件，这类文件包含了代码和数据，可以被用来链接成可执行文件或共享目标文件。linux系统的.o文件就是可重定位文件，对应Windows系统的就是.obj文件。这种可重定位文件也叫静态链接库，在之后的链接成的可执行文件中，都有一个代码的拷贝，如果很多可执行文件都使用同一个静态链接库，那么等于说每一个可执行文件都包含了这个静态链接库里的代码。
6. 第三种是共享目标文件。这种文件也是包含了代码和数据。但是和第一种可重定位文件不同，这个文件在不会最终存在于可执行文件中，而是以单独的文件形式存在，程序运行的时候，会链接到这个文件。同时这个文件可以可以多文件公用的，故不用占用大量的磁盘空间。也叫做动态链接库。对应的是Linux中的.so文件，以及Windows中的DLL文件。
7. 第四种是核心转储文件，是在进程以外终止时，将该进程地址空间的内容和终止时的一切信息存储到的文件。常用的就是上面四种格式文件。
8. 这是ELF文件的一个总结。很重要，考试要考的。
9. ELF这一标准，首次出现在1988年发布的UNIX System V Release4 操作系统，简称SVR4。由于表现好，在1999年的时候被x86open项目中Unix及类Unix操作系统中的标准文件格式。
10. 一个标准的ELF文件可以分为四个部分。第一部分是ELF header，用来描述整个文件。第二部分是Program Header Table，用来描述文件中的各种segments，告知系统如何创建进程。第三部分是sections或者segments。其中segments是从运行的角度来描述elf文件，sections是从链接的角度来描述elf文件。同时从图中我们可以看到，每一个segment是有多个sections所组成的，两者是包含关系。第四部分是Section Header Table，包含了每个section的信息。接下来我们针对这四部分，分别进行展开，仔细的探究一下ELF的组成。
11. 这里我借用网上的一张图，我认为这个是非常经典的，也是非常明了的一张图。事无巨细的分析了每一个组成部分。我会把这里拆开来给大家看。
12. 前面说了elf的四大组成部分，第一第二部分、划分为上面这个HEADER，分别是ELF的信息，以及Program Header Table。紧接着下面是sections，也就是主要存放数据的地方。这个sections拆开了，可以看到存放着代码、数据、以及section的名字。最后一部分是每个section的Header Table。
13. 这里 展示的彩色代码，就是一个真实文件的16进制值表示。上面是ELF header，下面是Program header table。（点击鼠标）来展开看一下。这里的彩色主要是为了进行每一个变量的区分。一共是十四个变量。
    1. 前面四个字节是固定，称之为“魔数”，里面包含了ELF字样。
    2. 后面两个绿的分别表示了文件是多少位的，01表示32位。
    3. 后面的表示文件是大端还是小端，01表示是小端。
    4. 再后面四个黄色的字节，永远是1，表明ELF的版本。
    5. 然后到第二行，e\_type，2表示的这是一个可执行文件，
    6. 蓝色的28表示的是ARM处理器。
    7. 再往后四字节绿色的和上面的一样，表示ELF版本，永远是1。
    8. 下面黄色的四位是一个地址，表明从哪里开始是可执行文件。
    9. 后面红色的16进制值0x40表示的是Program header的偏移值。
    10. 来到第三行，第一个蓝色十六进制值0xB0表明的是Section header 的偏移值。
    11. 后面绿色的0x34表明的是ELF header的大小。
    12. 后面黄色的0x20表明的是单个program header的大小。
    13. 后面红色的0x01表明现有的program header数量。
    14. 最后蓝色的0x28是说一个section header有多大。
    15. 第四行，绿色的4表明section header的总数量。
    16. 后面黄色的3表示names sections在table中的序号。这些就是ELF header的所有内容，可以看到就是一些基本的文件信息。
14. 下面的这个模块就是 Program Header Table的内容。这一块数据在什么位置呢？前面已经给出了，偏移量是0x40。第一行蓝色的1表明这一个segment需要载入到内存。
15. 绿色的0表示，这个segment应该加载到哪里，是一个偏移量，也是0。
16. 黄色是它应被载入到的虚拟地址
17. 红色是它应被载入到的物理地址。
18. 第二行绿色的90是文件的大小
19. 黄色的90是在内存中的大小
20. 红色的05表示文件即可读也可以执行。
21. 第二部分，就是Section，也就是ELF文件实质的内容。最上面一块是代码，中间是数据，最后是Section names。这一份示例文件就是一个输出Hello World的简单演示程序，可以看到，左边的十六进制代码翻译过来就是汇编语言编写的程序。中间存储的是一个字符串，Hello World。最后一个部分是section names。
22. 第三部分，Section Header Table。这一部分只是在链接的过程中会用到。对于可执行文件，是被忽略掉的。这里包含了name，type，flags，addr等字段。比较重要的字段，.text 是代码段，权限为只读，保存程序代码。.data数据段，保存已经初始化的全部变量和局部变量，可读可写。这里的.rodata就是read only，只读数据段。.shstrtab意思是section header string table。
23. 了解了这些信息之后，我们就对ELF文件整体的内容有了一个概念。在Linux系统下，有专门的命令来查看文件的elf文件头信息。比如readelf -h file。这里看到的就是我前面给大家讲的每一块的内容，这里是软件很直观的给大家展示出来。
24. 刚才图片中示例程序的段表展示了三种不同的段，但其实elf中存在更多不同种类的段，查看方法也很简单 readelf -S file。这张图里可以看到，除了序号0为留空以外，这个文件还包含另外九种不同的段。每一个不同的段都用来保存不同种类的数据。
25. 段表的作用就是用来描述ELF文件中各个段的信息，比如每个段的段名、段的长度、在文件中的位置、读写权限等。
26. 我们看一个简单地例子。非常简单的一个程序，交换两个变量。这里的bss段是用来保存未初始化的全局变量。已经初始化的全局变量存放在data段。text段用来存放代码。
27. ELF文件中有很多字符串，比如段名、变量名、函数名等。这些字符串并不保存在代码段，而是集中保存在字符串表中，便于查询。下面图中就是一个简单字符串的存储
28. 对应的还有符号表。函数和变量统称为符号，其中函数名和变量名就称为符号名。符号表保存了当前ELF文件中的所有符号的信息，包括符号名、符号对应的值、符号大小、符号类型、符号类型及绑定信息、符号所在的段。
29. 这里是ELF符号的结构体。
30. 查看符号表的命令是readelf -s file。我们看一下一些符号。
32. Linux的内核文件是要加载到内存中的。通过编译我们能得到很多种不同的内核映像文件。第一种vmlinux文件，属于是未经压缩，原始的映像。另一种是经过压缩的zImage文件，这类文件体积小，但是需要在映像开头添加解压缩的代码。每次载入的时候进行解压，由于体积小，比较适合嵌入式的系统。bzImage的b是big的意思，大。实际使用中两种映像分别载入到低端内存和高端内存区域。
33. 我们打开linux的/boot目录，就可以看到内核文件。这里展示了多个不同版本的内核文件。
34. 使用readelf命令就可以查看内核的内容了。如果不是elf格式的话，是无法通过readelf文件来查看的。该命令会通过文件初始的Magic number来判断当前文件格式。下面的一些基本信息，和我们前面分析的elf内容都是一一对应的。
35. objdump文件是Linux下面的命令，可以对文件进行反汇编。比如objdump -h file就可以查看目标文件的Section Header信息。内核影响包含了非常多的节，这里只展示了一小部分。
36. 回到boot目录，system.map文件是内核符号表，是从vmlinux中导出的。
37. 符号有很多种不同的类型，左边的列表详细的说明了符号类型，右边是符号表的格式，分别是地址、类型以及符号。小写的符号代表局部，大写的符号代表全局。
39. 加载到内存中的内核分为了两部分，一部分存储在640k一下的内存中，以实模式运行。另一部分存储在1MB以后的内存中，以保护模式运行。