**17373197 胡柯洋**

**一、实验思考题**

Thinking 1.1

**也许你会发现我们的readelf程序是不能解析之前生成的内核文件(内核文件是可执行文件)的，而我们之后将要介绍的工具readelf则可以解析，这是为什么呢？(提示：尝试使用readelf -h，观察不同)**

我们自己写的readelf.c文件在从Ehdr结构体中取出相应数据时默认ELF文件是小端存储的格式。./readelf文件夹中的testELF文件也是小端存储的格式。但是ELF文件的编码格式有大头编码与小头编码，用readelf-h观察之前生成的vmlinux文件时，可以发现之前生成的内核文件属于大头编码，在从中取出的多字节的数据中，“权值”比较大的高位数位于低字节，我们的readelf程序中需要增加对大小头编码的判断，以及对大头编码格式的ELF文件的格式进行处理，才能解析之前生成的内核文件。

Thinking 1.2

**main 函数在什么地方？我们又是怎么跨文件调用函数的呢？**

main函数位于init文件夹下的main.c，调用init.c中的mips\_init函数。

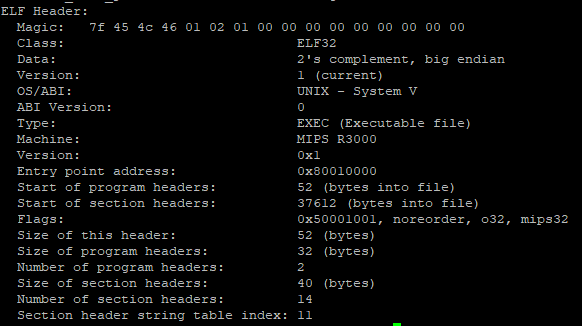
通过makefile中的link\_sript以及object两个宏定义，在编译时将tools文件夹下的.lds文件与boot、init文件夹下的.o、.c文件链接起来。.lds文件负责将内核加载到正确位置。

**二、实验难点图示**

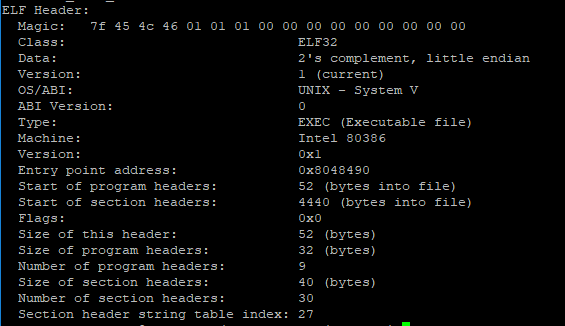
1.ELF文件数据保存模式大端与小端的差异

使用readelf -h 分别查看vmlinux与testELF

vmlinux:

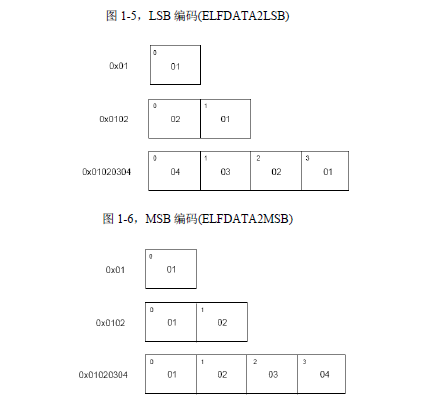


testELF:



不难看出，二者在Data一项中，vmlinux为big endian，testELF为little endian；

在课上测试时，由于个人的紧张以及时间的紧凑，没能写出好的大小端数据处理代码，通过对ELF手册的查阅，可以看出，MSB与LSB存储格式的差异



再通过查阅32位数据类型表



可知，处理vmlinux类似的大端存储类型的数据时，可以通过位运算并将结果按位与再按位或的方法，将4字节或2字节的数据由大端存储转化为小端存储。在我们自行书写的readelf.c文件中增加对大小端的处理，就完成对这两种格式的兼容。

**三、体会与感想**

我觉得这次lab1实验难度绝对是有的，而且绝不是指导书上说的难度较为简单。可能这是我们第一次自行阅读一个工程文件，对于各种文件的寻找还是磕磕绊绊的。对于大端与小端两种格式的ELF文件的读取，在课上测试时我很快找到了这两种格式的差异，但是对于大端小端转化这个步骤我采用了u\_char数组的方法，调了半天没调出来。可见操作系统课程对于C语言、汇编等的基础知识的要求还是很高的，我们需要在基础技能上多下一点功夫，在需要使用时才能游刃有余。

对于printf的书写，也要求我们对printf的输入输出格式较为熟悉，虽然认真阅读print.c的输出函数后我已经很好地理解了功能作用，但回想起刚读lib文件夹下文件时的一片空白，还是心有余悸。