Lab1实验报告

19376273 陈厚伦

思考题

Thinking1.1

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，而我们刚才介绍的工具 readelf 则可以解析，这是为什么呢？(提示：尝试使用 readelf -h，观察不同)

当我使用./readelf vmlinux命令时弹出"段错误 （核心已转储）"信息，我尝试使用readelf -h *命令观察不同的ELF文件的信息，发现是数据编码格式导致的问题。观察Magic[5]可以看到vmlinux是MSB编码，而其余可以解析的文件则是LSB，导致读取信息时发生错误，例如0x01020304被读成了0x04030201.在lab1课上测试中我完成了trans()函数完成了对vmlinux文件的解读。

```
(vmlinux)Magic:   7f 45 4c 46 01 02 01 00 00 00 00 00 00 00 00 00
```

Thinking1.2

内核入口在什么地方？main 函数在什么地方？我们是怎么让内核进入到想要的 main 函数的呢？又是怎么进行跨文件调用函数的呢？

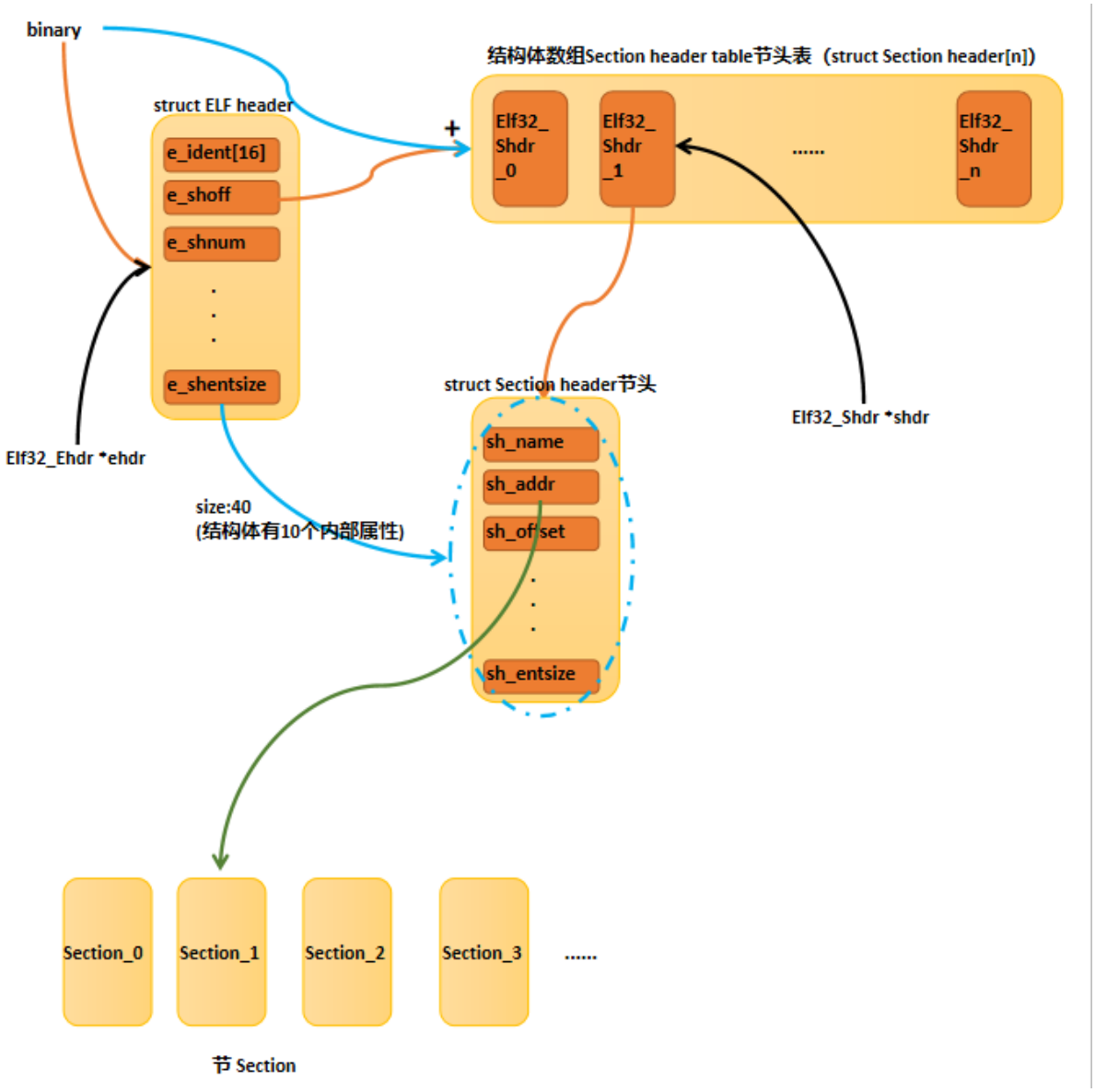
内核入口是`_start`函数在`boot/start.S`，`main`函数在`init/main.c`，内核启动先执行`_start`入口函数，然后从这个函数跳转到`main`函数，从`start.S`文件中`jal main`命令可以看出。这样内核启动的入口地址就可以固定下来，只需要传递main函数的地址就可以实现不同位置的main函数的调用。

跨文件调用函数：通过include头文件的方式。将需要函数写入.h文件，然后其他文件需要使用这些函数时include相应头文件即可。

实验难点图示

> ELF文件的结构——ELF文件的解析

ELF文件在编译链接和运行两种视角下的结构，指导书已经解释地很明确了，这里就不赘述了。在实验过程中关键要理解ELF文件中的结构体，事实上ELF文件的解读是我们要用正确的结构体变量在正确的位置找到其属性，这里需要明确一下这些结构体之间的层次关系，以本次联系用到的Section header为例，我绘制了不同层次结构的关系：



要注意选择正确的结构体指针，不同的结构体指针可能指向同一个地址值，但是其含义不同，其颗粒度也不同（p++运算的结果不同）

在对指针的地址值进行运算的时候要注意到指针颗粒度的问题，练习中要求我们找到节头表的首地址，由于e_shoff是以字节（8位）为单位存储的，所以在计算地址的时候我们要考虑到这一点，所以使用`unsigned char*`类型指针进行地址的运算，得到节头表的地址后，我们要将节头表按照节头数组的形式访问，所以此时我们访问的单位是节头结构体，所以我们要将地址的指针类型进行转换。

> 内核地址与Linker Script

在`mmp.h`文件中给出了结构图，在这个图中可以找到kernel部分的存储空间，所以我们的任务就是要重定位到指定的位置

第一步我们要设置静态确定的内存空间。通过Linker Script的SECTIONS部分将data、text、bss段的地址重定位。

第二部我们要考虑到动态运行时的内存空间，我们要事先确定这部分空间的范围。由于空间的上边界是栈，我们要在_start进入main函数时将sp指针指向stack的地址，于是内存空间的上边界也完成了重定位。

> printf与字符识别状态机

第一个难点是字符串的解析，由于题目要求比较简单，用状态机分析清晰易懂。

定义状态

0:start 已识别到**'%'**，开始解析格式字符串

1:flags 解析到**'0'**或**'-'**，开始解析flags部分

2:width 解析width

3:start_prec 解析到**'.'**，开始准备解析prec

4:prec 解析prec

5:length 解析length

6:switch 即将解析specifier，进入switch语句段

如下是状态转移表**S**，其中**s(i,j)**表示状态 **i** 到状态 **j** 的转移条件。

state	0	1	2	3	4	5	6
0		'0'|'-'	1~9	'.'		'l'	else
1			0~9	'.'		'l'	else
2			0~9	'.'		'l'	else
3					0~9	'l'	else
4					0~9	'l'	else
5							No condition
6							

第二个难点是利用辅助函数实现不同specifier下的输出，这里考验我们阅读别人代码的能力。此处d(D)情况下使用PrintNum()函数时出现了bug，阅读了PrintNum()函数后发现传递的num应当是绝对值，数字的正负由negFlag表示。所以在之后的lab中，除了自己写代码外，更要注重阅读源码。

体会与感想

由于lab0熟悉了linux操作环境，git版本管理工具，本次试验中可以说是相比lab0得心应手了，误删了重要文件后通过版本回退拯救了自己，通过tmux、vim内置命令、grep、gcc等命令是我能够非常舒服地写c代码与调试，工程技巧和经验积累量一些，并且能够自己写一些简单的Makefile、*.sh帮助自己，对Makefile中依赖这个概念有了更深入的了解。

整个过程依旧感到很痛苦，虽然大体明白要实现的目的与过程，但是对其中各种细节模模糊糊让我感觉很难下手，例如printf中设计出了自动机的算法，但是对其中的各种各样的指针，宏函数、辅助输出函数、可变长参数等新事物感到云里雾里，甚至不知道自己要用什么怎么用，所以发呆了很长时间。

虽然做完了练习题并得到了分数，但是依旧对整个过程感到很模糊。内核的启动过程，交叉编译等过程感到迷惑，各种术语不清楚，对其中各种目录下奇奇怪怪的文件不知道什么用途。但是实验结束后可以看出有很多其实只是要大致了解即可，**我应当培养的能力是从众多文件中知道哪些是我需要用到的，然后再研究这个范围内的细节**，当然也是要慢慢来。

lab1反复阅读指导书，总算弄清楚了ELF文件格式，对它有了一个形象化的认知。其中各种结构体的引用让我恶补了一番c语言指针的知识，也算是有所收获，最终做出了printf练习，有一定的成就感。

指导书反馈

建议增加一部分对专业术语、概念的讲解，网上能够查到的很多资料对初学者来看都是用一些专业名词解释另一些专业名词，让初学者很难找到学习的入口点，如果指导书对概念加一些解释会更好。

有些概念的汉语名称可以统一，或者一律用英文，比如program(segment) header table、section header table。

思考题和练习能否向讨论区一样单独开辟一个页面，方便自己查找自己是否有遗漏。

残留难点

Makefile的更多强大的功能还有待我解锁

学习如何在linux环境下搭建自动测评