

**课 程 实 验 报 告**

**课程名称： 计算机系统基础**

**实验名称： ELF文件与程序链接**

**院 系 ：计算机科学与技术**

**专业班级 ： 本硕博202101**

**学 号 ： U202115674**

**姓 名 ： 姚晨炫**

**指导教师 ： 许向阳**

**2022 年 11 月 7 日**

**一、实验目的与要求**

通过修改给定的可重定位的目标文件（链接炸弹），加深对可重定位目标文件格式、目标文件的生成、以及链接的理论知识的理解。

实验环境：Ubuntu，GCC，GDB等。

**二、实验内容**

**任务7.1** 链接炸弹的拆除

**在二进制层面，逐步修改构成目标程序“linkbomb”的多个二进制模块（“.o文件”），然后链接生成可执行程序，要求可执行程序运行能得到指定的效果。修改目标包括可重定位目标文件中的数据、机器指令、重定位记录等。**

**提供的实验包含有 main.c、phase1.o、phase2.o、phase3.o 等。生成执行程序 linkbomb 的方法：**

**#gcc -no-pie -o linkbomb[n] main.o phase[n]\*.o n=1、2、3……**

**使用 readelf、hexdump、od、hexedit 等工具阅读和修改目标文件。**

**1、第1关 静态数据与ELF数据节**

修改二进制可重定位目标文件 phase1.o 的数据节中的内容（不允许修改其他节），使其与main.o链接后，生成的执行程序，可以输出自己的学号。

**2、第2关 简单的机器指令修改**

修改二进制可重定位目标文件 phase2.o 的代码节中的内容（不允许修改其他节），使其与main.o链接后，生成的执行程序。在phase\_2.c 中，有一个静态函数 static void myfunc( ) ，要求在 do\_phase 函数中调用myfunc( )，显示相应的信息。

**3、第3关 有参数的函数调用的机器指令修改**

修改二进制可重定位目标文件 phase3.o 的代码节中的内容（不允许修改其他节），使其与main.o链接后，生成的执行程序。在phase\_3.c 中，有一个静态函数 static void myfunc(int offset) ，要求在 do\_phase 函数中调用myfunc(offset )，显示相应的信息。

**4、第4关 有局部变量的机器指令修改**

修改二进制可重定位目标文件 phase4.o 的代码节中的内容（不允许修改其他节），使其与main.o链接后，生成的执行程序。在phase\_4.c 中，有一个静态函数 static void myfunc(char \*s) ，要求在 do\_phase 函数中调用myfunc(s )，显示出自己的学号。

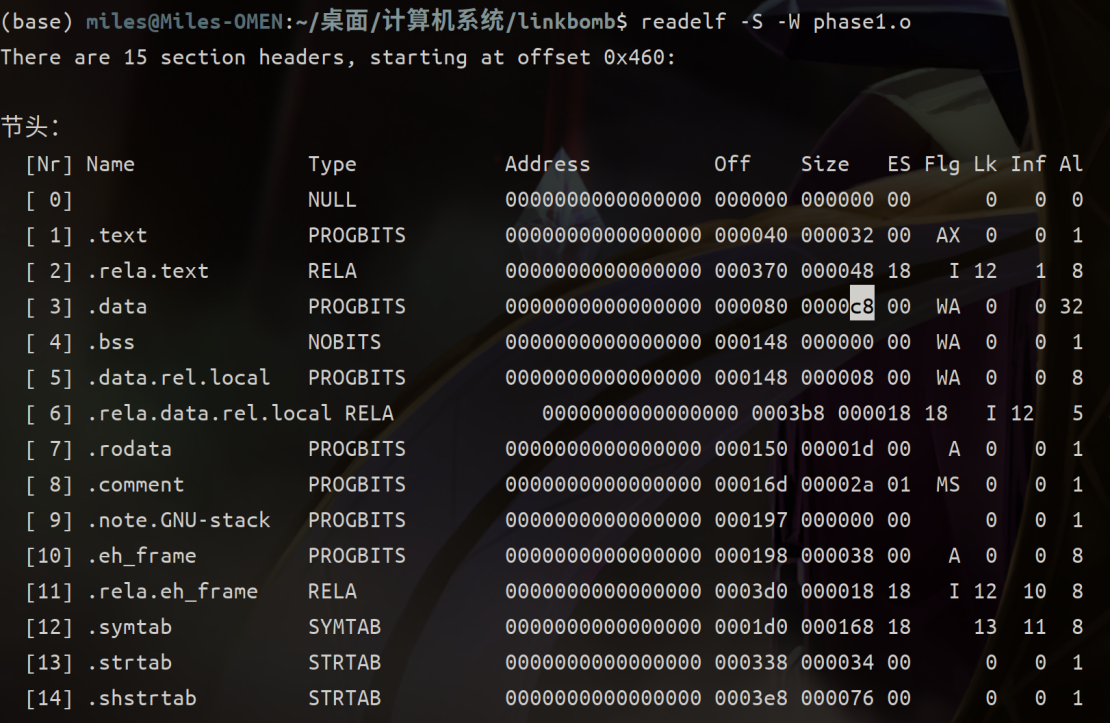
**5、第5关 重定位表的修改**

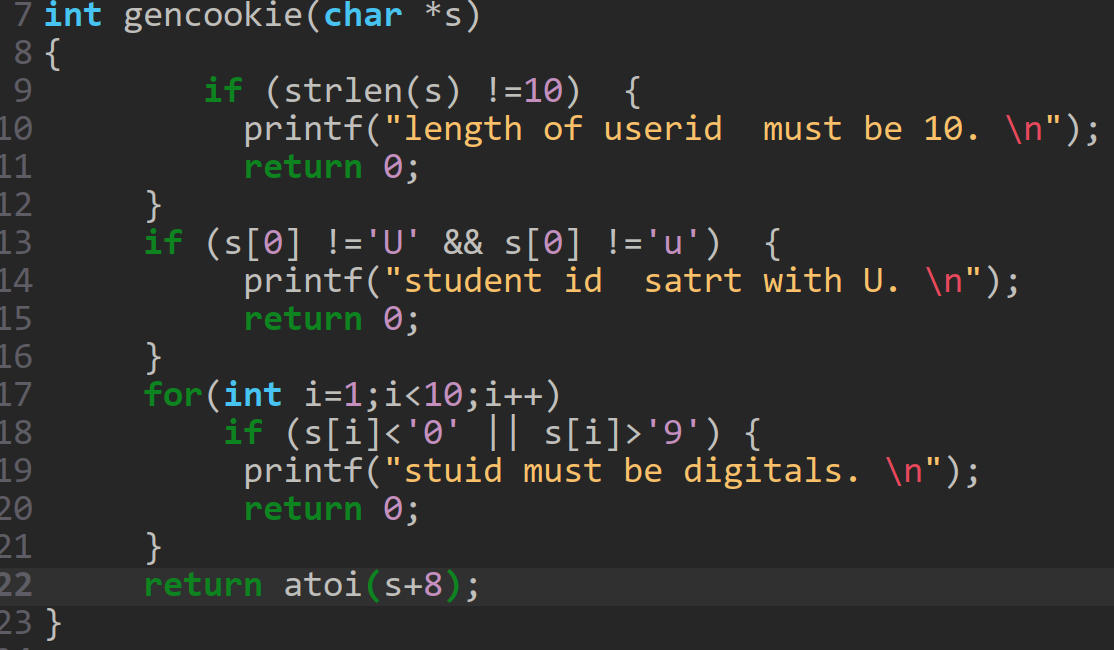
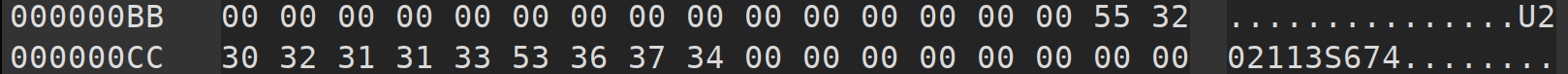
修改二进制可重定位目标文件 phase5.o 的重定位节中的内容（不允许修改代码节和数据节），使其与main.o链接后，生成的执行程序运行时，显示Class Name : Computer Foundation. Teacher Name : Xu Xiangyang。

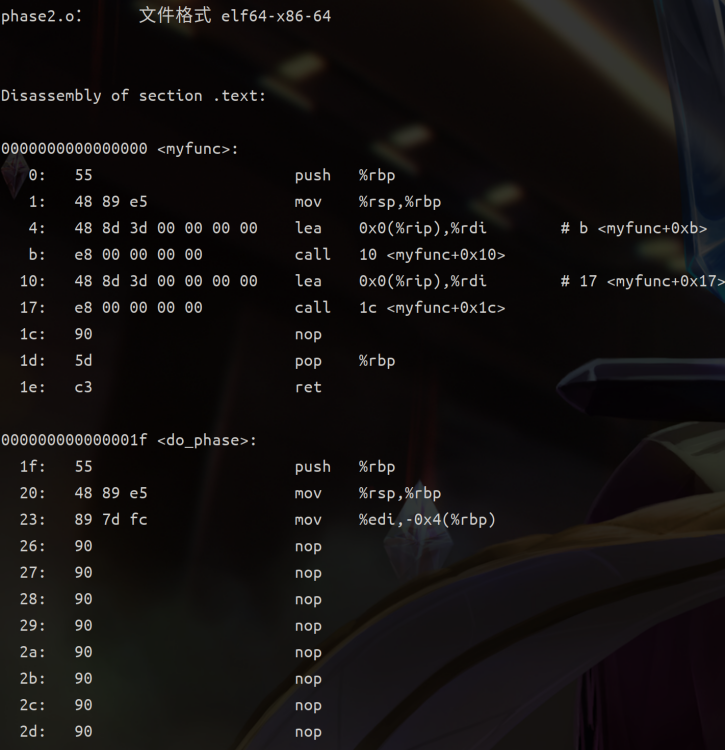
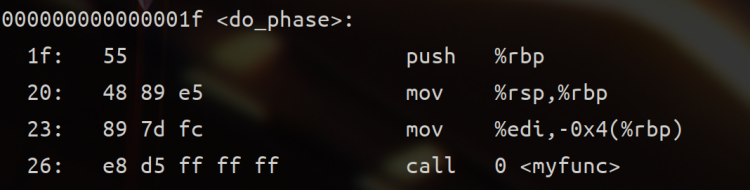
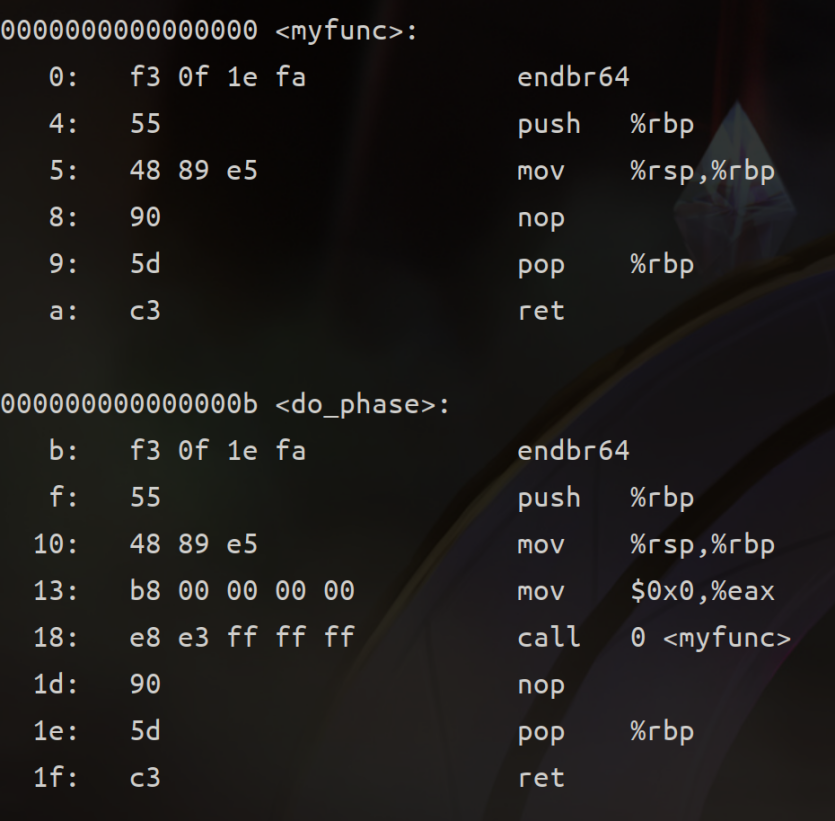
**三、实验记录及问题回答**

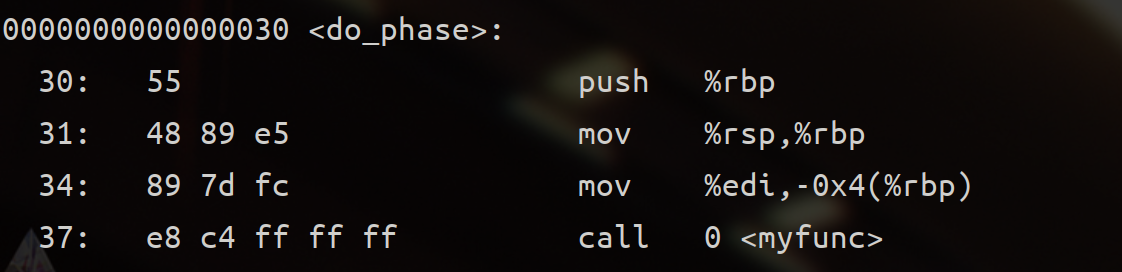
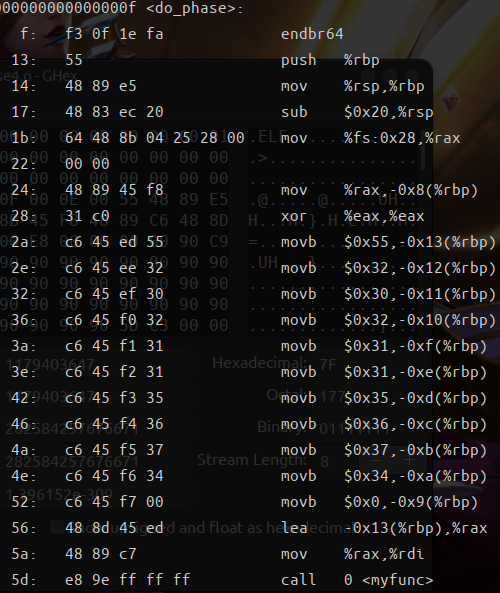
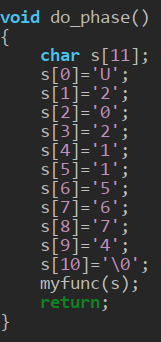
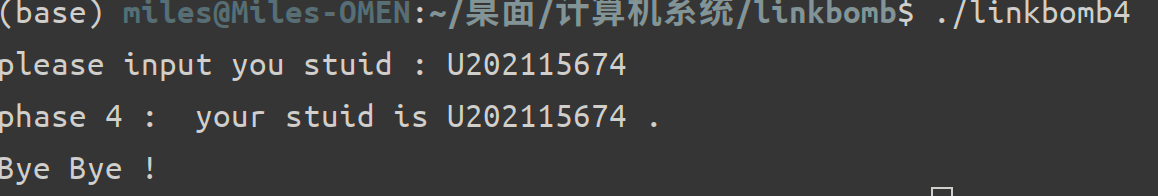
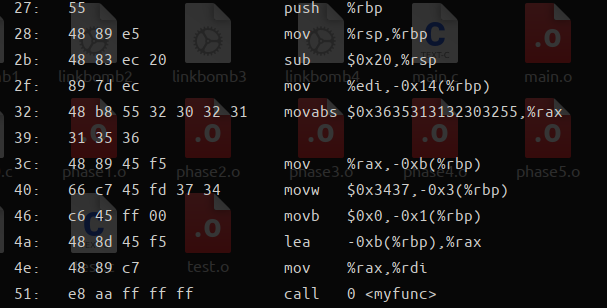
**（1）实验任务 7.1 的实验记录**

1.

先使用readelf读取各个节的地址然后找到数据节的地址，再观察phase1的反汇编代码，

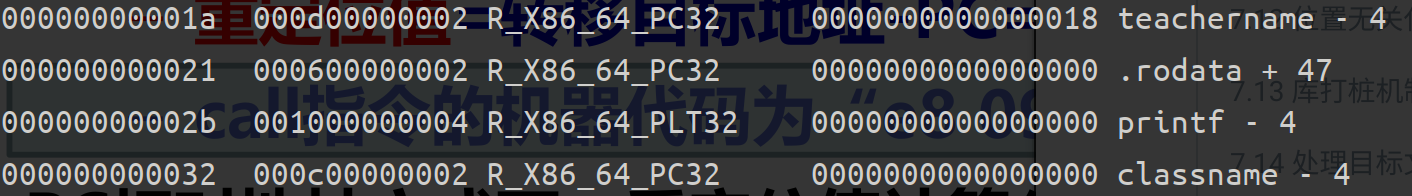
发现程序在do\_phase函数中设置了一个偏移量，是基于学号生成的，大小为学号的两位，相关程序如下图所示：。对应的偏移为4a+80=ca，然后我们是Ghex找到对应的位置，将对应位置的数据值设置为自己的学号

观察phase2的反汇编代码，我们希望在26行的地方加入一个跳转指令，去调用myfunc函数，所以我们仿照着写一个c语言程序，去调用myfunc函数，然后观察他的机器码。可以看到call指令的机器码为e8 d5 ff ff，其中d5 ff ff ff为补码形式，是基于下一条指令的向前跳转。所以我们找到phase5中的text节，需要向前跳转1d个字节，所以在对应的位置填入e8 e3 ff ff ff 反汇编之后的代码如图，可见程序成功的跳转到myfunc的位置。

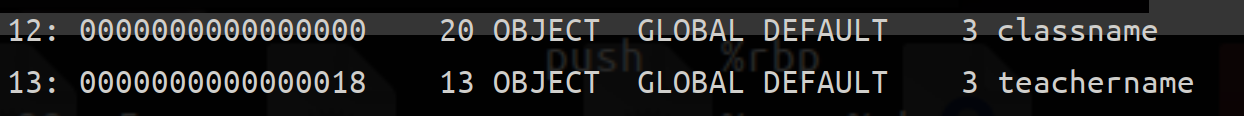
1. phase3与phase2的区别在于phase3中需要传递参数，我们要将do phase中的参数继续传递到my func中，但是由于x86体系下参数是使用寄存器传递的，因此在进入do phase函数之后edi的值未被改变，我们可以直接继续调用myfunc函数，这样do phase的参数被继续传递到了myfunc中。因此第三关的代码与上一关是相同的。
2. phase4希望我们将自己的学号打印出来，因此我们的想法是将学号放入栈中，然后使用参数的传递将我们的数组地址传入myfunc中，所以我编写了一个将学号放入栈中的函数，如下我的想法原来就是直接将这一段代码放入phase5中，然后就可以了，但是phase5中的.text节的长度是有限的，如果使用这种方式会将后面的数据覆盖，但是插入新的字节会破坏原来的地址关系。所以我采用了下图所示的方法，一次传入八个字节，减少了.texxt节的占用，写入phase5,通过了这关。

5.

这一关要求我们调换teacher和classname的位置，我们的想法是在重定位的时候我们将teacher定位到classname的位置，然后再将classname定位到teacher的位置。使用readelf -r 来查看重定位节。观察重定位节可以看到teacher在节中的偏移地址为1a，classnmae为32,然后对应的符号表的编号分别为d和c。



使用objdump查看符号表中可以看到确实对应的符号表节为12和13.因此我们进入hex中，讲teacher的重定位对应的符号信息改为classname的，将classname的改为teacher的，即将c和d互换，就通过了这一关。



**四、体会**

这是本学期的最后一次实验，在实验室我学会阅读了.o文件中不同节的相关内容，掌握了重定位的相关知识。还学会了如何在2进制的维度对不同节进行相关的查看，理解和修改。在做到第四关的时候，还学会了对汇编语言进行相关的简化，不使用重复大断的机器语言来写入数据。在第五次实验中学会了修改重定位节来改变相关参数的位置从而实现参数的调换，让我对重定位过程有了更深的理解。