

**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 ： U202115666**

**姓 名 ： 刘文博**

**指导教师 ： 许向阳**

**2022 年 10 月 1 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。

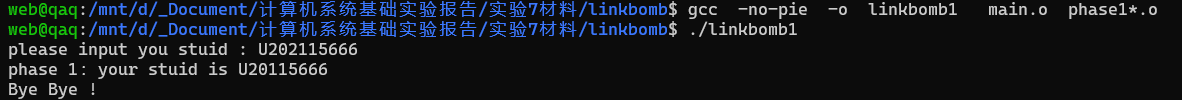
**三、实验记录及问题回答**

**（1）实验任务 7.1 的实验记录**

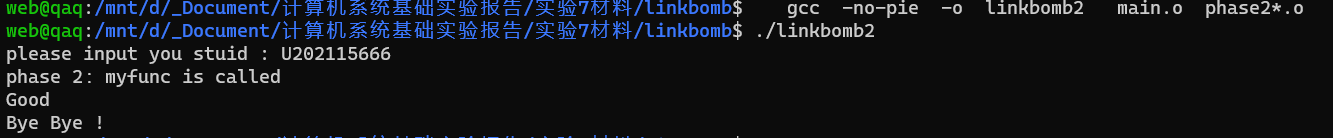
用sudo apt-get install hexedit安装hexedit

首先使用gcc -c main.c -o main.o 生成main.o文件之后用 gcc -no-pie -o linkbomb（x） main.o phase(x)\*.o 生成linkbomb(x) x=1,2,3,4,5

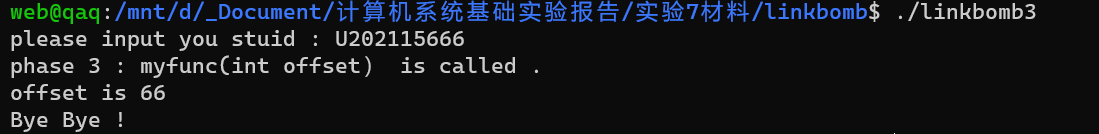
按照要求修改phase1.o的内容

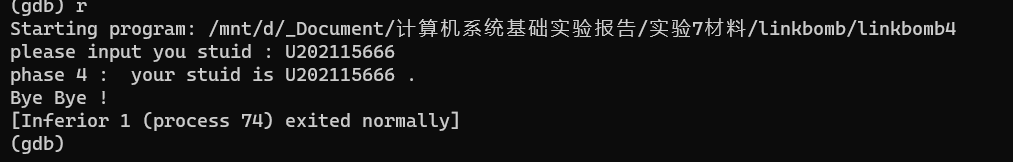


按要求修改phase2.o的内容

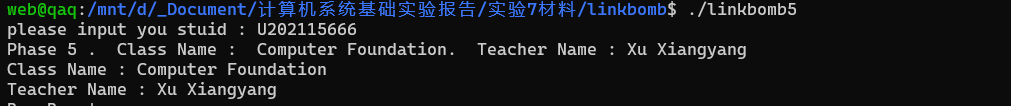


按要求修改phase3.o的内容

 按要求修改phase4.o的内容



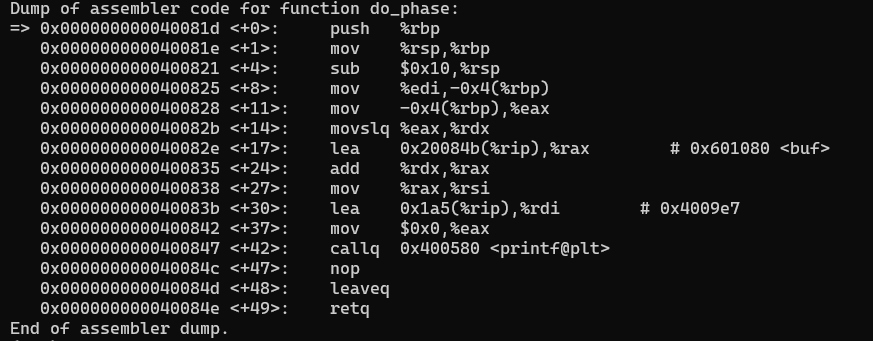
按要求修改phase5.o的内容



**（2） 描述修改各个文件的基本思想**

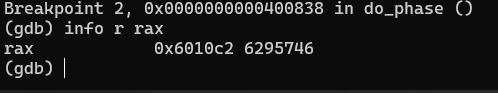
第一关

对do\_phase函数进行反汇编

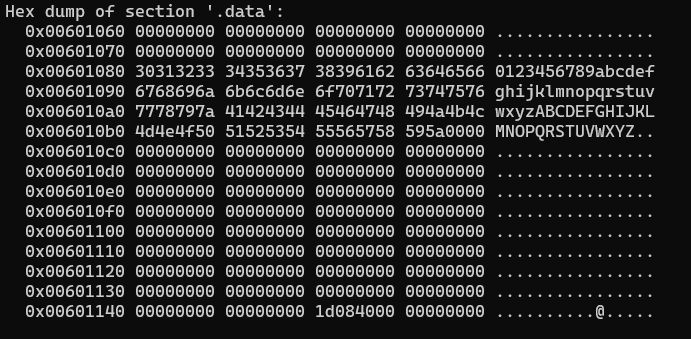


发现它会打印从buf首地址第rdx个元素开始的字符串。

查看运行到 0x0000000000400838 <+27>: mov %rax,%rsi指令时rax的值



查看linkbomb1数据段的内容

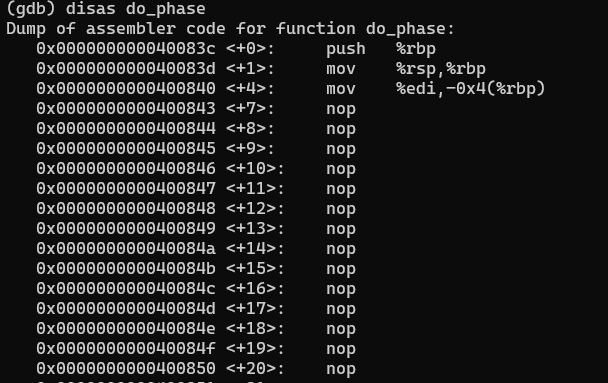


此时就可以知道要修改的内容位于哪里了，将其修改为学号的16进制表示。

重新生成可执行文件即可。

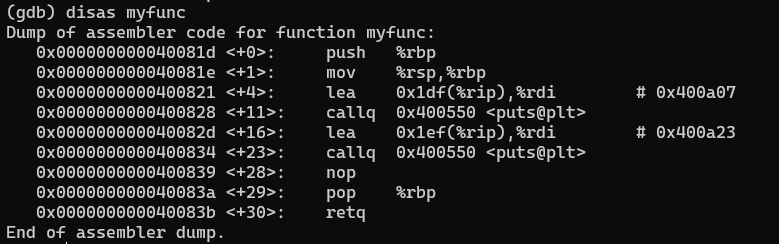
第二关

反汇编do\_phase函数发现它执行了很多空操作

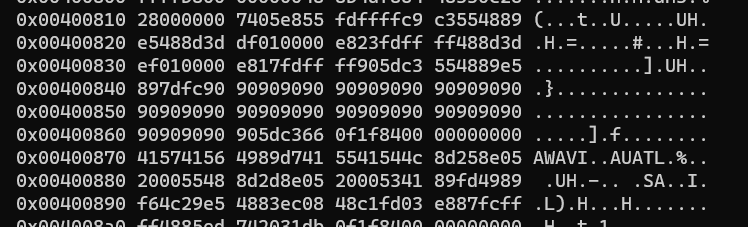


这样我们可以用新的指令顶替nop指令的位置

反汇编myfunc得到其入口地址



用readelf读取linkbomb2的text节内容，找到大段nop的位置



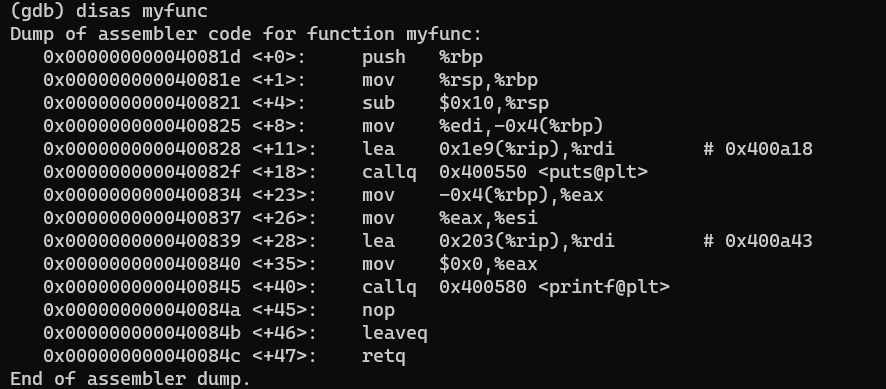
根据call指令的使用方法，我们可以在 0x0000000000400843 <+7>: nop

处将其修改为call myfunc，这时候其机器码为e8 0x000000000040081d-0x0000000000400843+5（call指令占5个字节）=0xffffffffffffffd5



修改后重新生成linkbomb2即可

第三关

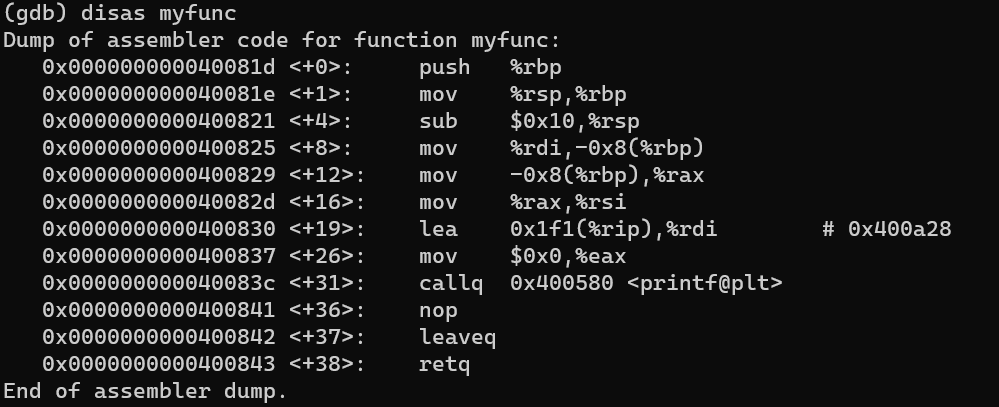
反汇编myfunc（int offset）

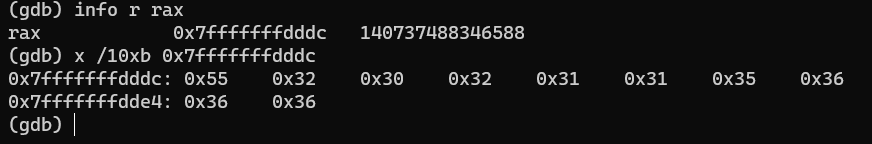
发现edi中应该保存着int offset，rdi中保存着输出格式。

故在do\_phase中加入call myfunc 这两条指令。与上一关类似。对应着0x000000000040081d- 0x0000000000400859=0xffffffffffffffc4

修改后重新生成linkbomb2即可

第四关

反汇编myfunc，容易发现一开始rdi为保存着传入的参数也就是学号的首地址

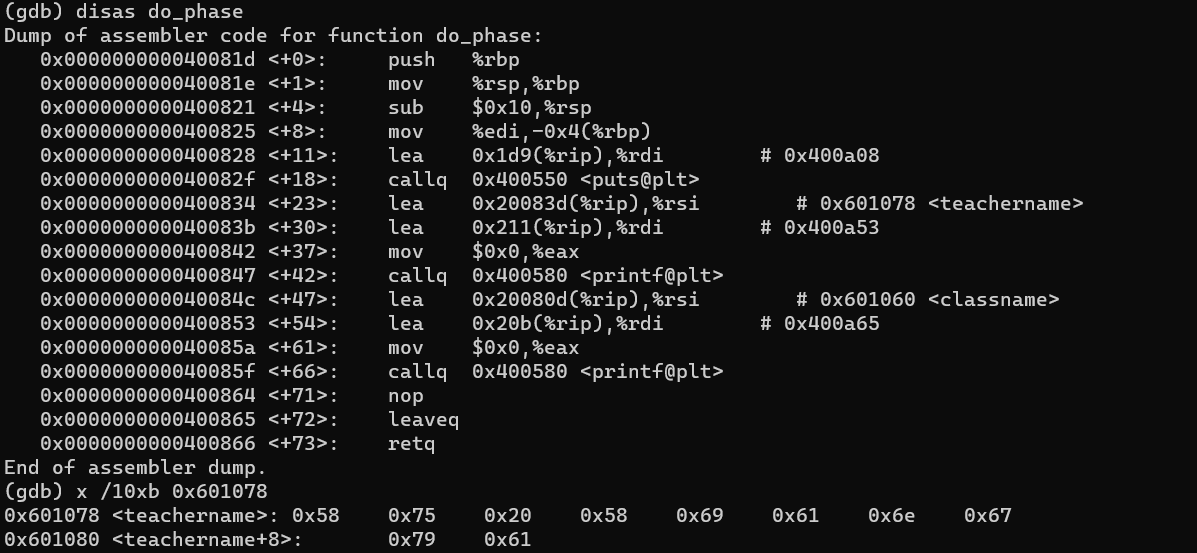


对main函数进行反汇编再通过反汇编info指令得知学号以0x7fffffffdddc为首地址，所以需要用 lea 0x7fffffffdddc，rdi 和call myfunc 来替换nop。

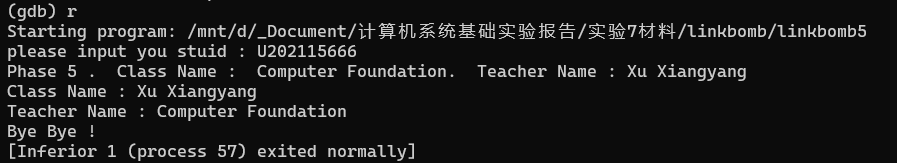
鉴于知识水平有限，只找到了movabs $0x7fffffffdddc,%rax 和 mov %rax,%rdi的机器指令借此间接达到lea 0x7fffffffdddc，rdi的目的。对应的机器指令为48 b8 dc dd ff ff ff 7f 00 00 48 89 c7 e8 c0 ff ff ff 。修改之后重新链接即可。

第五关

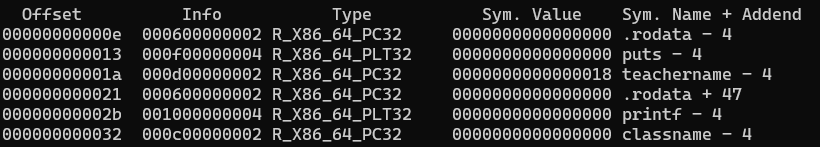
反汇编do\_phase,发现函数应该是可以正确打印老师和课堂名的。



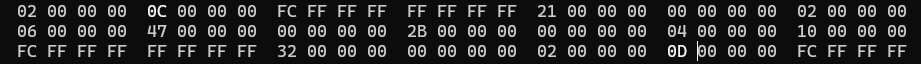
但实际运行时发现 第二次输出时class name和teacher name出现错位，将二者重定位信息颠倒可以纠正这一错误。



查看重定位信息



根据重定位信息中的内容在hexedit中找到相应位置再将二者对应的符号信息（info）对调。最后重新链接即可。



**四、体会**

经过链接炸弹的破解，我对节的概念和如何修改相应节的内容有了更深刻的认识，掌握了根据偏移地址计算实际地址的方法。明白了函数压栈的内容在其运行中的作用。总而言之，在破解炸弹的过程中我对程序的链接有了更好的认识。