

华中科技大学

# 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： CS2304

学 号： U202315653

姓 名： 郝依凝

指导教师： 金良海

2025 年 4 月 22 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

工具：GCC、GDB、readelf、hexdump、hexedit、od 等。

## 二、实验内容

### 任务 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。

#### 1、第 1 关 数据节的修改

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()，显示信息 myfunc is called. Good!。

#### 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(pos)，将 do\_phase 的参数 pos 直接传递 myfunc，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。

#### 6、第 6 关 强弱符号

不准修改 main.c 和 phase6.o，通过增补一个文件，使得程序链接后，能够输出自己的学号。

```
#gcc -no-pie -o linkbomb6 main.o phase6.o phase6_patch.o
```

#### 7、第 7 关 只读数据节的修改

修改 phase7.o 中只读数据节（不准修改代码节），使其与 main.o 链接后，能够输出自己的学号。

## 三、实验记录及问题回答

### (1) 实验结果及操作过程记录

```
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -no-pie -o linkbomb1 main.o phase1.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb1
please input you stuid : U202315653
your ID is : U202315653
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 1 第 1 关结果

```
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ hexedit phase2.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -no-pie -o linkbomb2 main.o phase2.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb2
please input you stuid : U202315653
myfunc is called. Good!
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 2 第 2 关结果

```
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ hexedit phase3.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -no-pie -o linkbomb3 main.o phase3.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb3
please input you stuid : U202315653
gate 3: offset is : 8!
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 3 第 3 关结果

```
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ hexedit phase4.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -no-pie -o linkbomb4 main.o phase4.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb4
please input you stuid : U202315653
gate 4: your ID is : U202315653!
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 4 第 4 关结果

```
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb5
please input you stuid : U202315653
Class Name Computer Foundation
Teacher Name Xu Xiangyang
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 5 第 5 关结果

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ code phase6_patch.c
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -c -g phase6_patch.c -o phase6_patch.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -no-pie -o linkbomb6 main.o phase6.o phase6_patch.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb6
please input you stuid : U202315653
U202315653
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 6 第 6 关结果

```
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ hexedit phase7.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -no-pie -o linkbomb7 main.o phase7.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb7
please input you stuid : U202315653
Gate 7: U202315653
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 7 第 7 关结果

## (2) 描述修改各个文件的基本思想

### 1) 第 1 关 数据节的修改

观察 do\_phase 函数反汇编代码：通过寄存器 edi 传入参数并将参数保存在寄存器 eax 中，通过 cltq 指令将 eax 扩展至 rax；lea 0x0(%rip), %rdx 指令获取输出字符串首地址，add %rdx, %rax 指令获取字符串打印起始位置。

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
• gabriel@gabriel-ThinkBook-16-65-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ objdump -d phase1.o

phase1.o: 文件格式 elf64-x86-64

Disassembly of section .text:

0000000000000000 <do_phase>:
0: f3 0f 1e fa          endbr64
4: 55                  push    %rbp
5: 48 89 e5            mov     %rsp,%rbp
8: 48 83 ec 10         sub     $0x10,%rsp
c: 89 7d fc            mov     %edi,-0x4(%rbp)
f: 8b 45 fc            mov     -0x4(%rbp),%eax
12: 48 98              cltq
14: 48 8d 15 00 00 00 00 lea     0x0(%rip),%rdx    # 1b <do_phase+0x1b>
1b: 48 01 d0            add     %rdx,%rax
1e: 48 89 c6            mov     %rax,%rsi
21: 48 8d 3d 00 00 00 00 lea     0x0(%rip),%rdi    # 28 <do_phase+0x28>
28: b8 00 00 00 00     mov     $0x0,%eax
2d: e8 00 00 00 00     call    32 <do_phase+0x32>
32: 90                  nop
33: c9                  leave
34: c3                  ret
  
```

图 8 phase1.o 代码节

观察 linkbomb1：验证 lea 0x0(%rip), %rdx 指令获取 buf 串首地址，通过 edi 传入参数作为偏移量，add %rdx, %rax 指令计算打印字符串首地址。

```

File Edit Selection View Go Run Terminal Help
实验5_链接炸弹
2023年_计算机系统基础_实验任务书_链接...
链接炸弹_显示提示.docx
linkbomb1
linkbomb2
C main.c
main.o
phase0.o
phase1.o
phase2.o
phase3.o
phase4.o
phase5.o
phase6.o
phase7.o

0x401385 <do_phase>          endbr64
0x401389 <do_phase+4>        push    %rbp
0x40138b <do_phase+5>        mov     %rsp,%rbp
0x40138d <do_phase+6>        sub     $0x10,%rsp
0x401391 <do_phase+12>       mov     %edi,-0x4(%rbp)
0x401394 <do_phase+15>       mov     -0x4(%rbp),%eax
0x401397 <do_phase+18>       cltq
0x401399 <do_phase+20>       lea     0x2cc0(%rip),%rdx    # 0x404060 <buf>
0x40139b <do_phase+27>       add     %rdx,%rax
0x40139d <do_phase+30>       mov     %rax,%rsi
0x40139e <do_phase+33>       lea     0x0(%rip),%rdi    # 0x402117
0x40139f <do_phase+40>       mov     $0x0,%eax
0x4013b2 <do_phase+45>       call    0x4010c0 <printf@plt>
0x4013b7 <do_phase+50>       nop
0x4013b8 <do_phase+51>       leave
0x4013b9 <do_phase+52>       ret
0x4013ba              add     %al,(%rax)
0x4013bc <_fini>             endbr64
0x4013be <_fini+4>           sub     $0x8,%rsp
0x4013c4 <_fini+8>           add     $0x8,%rsp
0x4013c8 <_fini+12>         ret
0x4013c9              add     %al,(%rax)
0x4013cb              add     %al,(%rax)
0x4013cd              add     %al,(%rax)
0x4013cf              add     %al,(%rax)
0x4013d1              add     %al,(%rax)
0x4013d3              add     %al,(%rax)
0x4013d5              add     %al,(%rax)
0x4013d7              add     %al,(%rax)
0x4013d9              add     %al,(%rax)
0x4013db              add     %al,(%rax)
0x4013dd              add     %al,(%rax)
0x4013df              add     %al,(%rax)
0x4013e1              add     %al,(%rax)

multi-thre Thread 0x7ffff7fa67 (asm) In: do_phase
L77 PC: 0x40138d

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0x7ffff7fc3800

[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Please input you stuid : U202315653
Breakpoint 1, 0x000000000040138d in do_phase ()
(gdb) x/4000 0x404060
0x404060 <buf>: 0x01 0x62 0x63 0x64 0x65 0x66 0x67 0x68
0x404068 <buf+8>: 0x55 0x32 0x30 0x32 0x33 0x31 0x35 0x36
0x404070 <buf+16>: 0x25 0x23 0x00 0x00 0x00 0x00 0x00 0x00
0x404078 <buf+24>: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x404080 <buf+32>: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
(gdb)
  
```

图 9 linkbomb1 调试

观察 main.c 文件：输入 stuid (U202315653)，通过函数 gencookie 获取 cookie 值—— $\text{cookie} = 5 + \text{atoi}(s + 9)$ ，即输入学号最后一位转为字符对应数字 (3) + 5 = 8。函数指针 phase 传入参数为 cookie，因此 do\_phase 函数传入参数为 8，即偏移量为 8。

```

5  extern void (*phase)(int i); // 定义了一个函数指针
6
7  int gencookie(char *s)
8  {
9      if (strlen(s) != 10) {
10         printf("length of userid must be 10. \n");
11         return 0;
12     }
13     if (s[0] != 'U' && s[0] != 'u') {
14         printf("student id satrt with U. \n");
15         return 0;
16     }
17     for(int i=1;i<10;i++)
18         if (s[i]<'0' || s[i]>'9') {
19             printf("stuid must be digitals. \n");
20             return 0;
21         }
22     return 5+atoi(s+9);
23 }
24
25 int main(int argc, const char *argv[])
26 {
27     int cookie;
28     char stuid[12];
29     printf("please input you stuid : ");
30     scanf("%s",stuid);
31     cookie = gencookie(stuid);
32
33     if (phase)
34         (*phase)(cookie);
35     else {
36         printf("Welcome to linkbomb \n");
37         printf("You should modify phase1.o, phase2.o ....\n");
38         printf("execute : gcc -no-pie -o linkbomb1 main.o phase1.o \n");
39         printf("execute : ./linkbomb1 \n");
40     }
41     printf("Bye Bye !\n");
42     return 0;
43 }

```

图 10 main.c 文件

观察节表头：数据节偏移值为 0x80，大小为 0x28。

```

gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ readelf -S phase1.o
There are 16 section headers, starting at offset 0x408:

节头:
[ 0] 名称      类型      地址      偏移量
     大小      全体大小  旗标  链接  信息  对齐
[ 0] 0000000000000000 NULL      0000000000000000 0 0 0
[ 1] .text      PROGBITS 0000000000000000 00000040
     0000000000000035 0000000000000000 AX 0 0 1
[ 2] .rela.text RELA      0000000000000000 00000300
     0000000000000048 0000000000000018 I 13 1 8
[ 3] .data      PROGBITS 0000000000000000 00000080
     0000000000000028 0000000000000000 WA 0 0 32
[ 4] .bss       NOBITS   0000000000000000 000000a8
     0000000000000000 0000000000000000 WA 0 0 1
[ 5] .data.rel.local PROGBITS 0000000000000000 000000a8
     0000000000000008 0000000000000000 WA 0 0 8
[ 6] .rela.data.r[...] RELA      0000000000000000 00000348
     0000000000000018 0000000000000018 I 13 5 8
[ 7] .rodata    PROGBITS 0000000000000000 000000b0
     0000000000000011 0000000000000000 A 0 0 1
[ 8] .comment   PROGBITS 0000000000000000 000000c1
     000000000000002c 0000000000000001 MS 0 0 1
[ 9] .note.GNU-stack PROGBITS 0000000000000000 000000ed
     0000000000000000 0000000000000000 0 0 1
[10] .note.gnu.pr[...] NOTE      0000000000000000 000000f0
     0000000000000020 0000000000000000 A 0 0 8
[11] .eh_frame  PROGBITS 0000000000000000 00000110
     0000000000000038 0000000000000000 A 0 0 8
[12] .rela.eh frame RELA      0000000000000000 00000360
     0000000000000018 0000000000000018 I 13 11 8
[13] .symtab    SYMTAB   0000000000000000 00000148
     0000000000000180 0000000000000018 14 11 8
[14] .strtab    STRTAB   0000000000000000 000002c8
     0000000000000034 0000000000000000 0 0 1
[15] .shstrtab  STRTAB   0000000000000000 00000378
     0000000000000089 0000000000000000 0 0 1

```

图 11 phase1.o 节表头



由上述分析得到应从偏移量  $0x80+0x8=0x88$  处开始修改字符串

55 (U) 32 (2) 30 (0) 32 (2) 33 (3) 31 (1) 35 (5) 36 (6) 35 (5) 33 (3) 00

00000078	00 00 00 00	00 00 00 00	61 62 63 64	65 66 67 68	55 32 30 32	33 31 35 36	35 33 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....abcdefghU202315653.....
000000A0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	79 6F 75 72	20 49 44 20	69 73 20 3A	20 25 73 0A	00 00 47 43	43 3A 20 28	.....your ID is : %s...GCC: (
000000C8	55 62 75 6E	74 75 20 39	2E 34 2E 30	2D 31 75 62	75 6E 74 75	31 7E 32 30	2E 30 34 2E	32 29 20 39	2E 34 2E 30	00 00 00 00	Ubuntu 9.4.0~1ubuntu1~20.04.2) 9.4.0...

图 12 修改数据节

保存修改后，重新链接生成的执行程序即可输出自己的学号。

```
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb1
please input you stuid : U202315653
your ID is : ijklmnopqrstuvwxyz0123456789
Bye Bye !
```

图 13 修改前

```
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -no-pie -o linkbomb1 main.o phasel.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb1
please input you stuid : U202315653
your ID is : U202315653
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 14 修改后

## 2) 第 2 关 简单的机器指令修改

观察反汇编代码：在代码节偏移量为 22~34 为 nop，35~36do\_phase 退出函数在此之间填入指令达到在 do\_phase 函数中调用 myfunc()目标。由于 myfunc 是静态函数，可以在 phase2.o 文件中计算 call 指令下一条指令的地址到 myfunc 的字节距离，0-call 指令下一条指令的地址得到一有符号负数，填充 call 指令位移量。call 指令长度为 5，myfunc 起始地址的偏移量为 0，因此下一条指令到 myfunc 的字节距离为  $0x5 + 0x22 = 0x27$ ,  $0 - 0x27 = -0x27$ ，得到对应补码为 0xffff ffd9。call 指令操作码为 e8，地址采用小端存储，因此指令为 e8 d9 ff ff ff。

```

0000000000000000 <myfunc>:
 0: f3 0f 1e fa      endbr64
 4: 55               push  %rbp
 5: 48 89 e5         mov   %rsp,%rbp
 8: 48 8d 3d 00 00 00 00 lea    0x0(%rip),%rdi    # f <myfunc+0xf>
f: e8 00 00 00 00    call  14 <myfunc+0x14>
14: 90               nop
15: 5d               pop   %rbp
16: c3               ret

0000000000000017 <do_phase>:
17: f3 0f 1e fa      endbr64
1b: 55               push  %rbp
1c: 48 89 e5         mov   %rsp,%rbp
1f: 89 7d fc         mov   %edi,-0x4(%rbp)
22: 90               nop
23: 90               nop
24: 90               nop
25: 90               nop
26: 90               nop
27: 90               nop
28: 90               nop
29: 90               nop
2a: 90               nop
2b: 90               nop
2c: 90               nop
2d: 90               nop
2e: 90               nop
2f: 90               nop
30: 90               nop
31: 90               nop
32: 90               nop
33: 90               nop
34: 90               nop
35: 5d               pop   %rbp
36: c3               ret
    
```

图 15 phase2.o 代码节

观察节表头：代码节偏移值为 0x40。

节头:	名称	类型	地址	偏移量
[号]	名称	类型	地址	偏移量
[0]	大小	全体大小	旗标	链接 信息 对齐
[ 0]	0000000000000000	NULL	0000000000000000	0 0 0
[ 1]	.text	PROGBITS	0000000000000000	0 0 1
[ 2]	.rela.text	RELA	0000000000000000	0 0 1
[ 3]	.data	PROGBITS	0000000000000000	0 0 1
[ 4]	.bss	NOBITS	0000000000000000	0 0 1
[ 5]	.rodata	PROGBITS	0000000000000000	0 0 1
[ 6]	.data.rel.local	PROGBITS	0000000000000000	0 0 1
[ 7]	.rela.data.r[...]	RELA	0000000000000000	0 0 1
[ 8]	.comment	PROGBITS	0000000000000000	0 0 1
[ 9]	.note.gnu-stack	PROGBITS	0000000000000000	0 0 1
[10]	.note.gnu.pr[...]	NOTE	0000000000000000	0 0 1
[11]	.eh_frame	PROGBITS	0000000000000000	0 0 1
[12]	.rela.eh_frame	RELA	0000000000000000	0 0 1
[13]	.symtab	SYMTAB	0000000000000000	0 0 1
[14]	.strtab	STRTAB	0000000000000000	0 0 1
[15]	.shstrtab	STRTAB	0000000000000000	0 0 1

图 16 phase2.o 节表头

由上述代码节中偏移 0x22 为空得到修改 0x40+0x22=0x62 处，添加 call 指令。

```
00000050  00 00 00 00 90 5D C3 F3 0F 1E FA 55 48 89 E5 89 7D FC E8 D9 FF FF FF 90 90 90 90 90 90 90 90 90 90 90 5D C3 6D .....UH...].m
```

图 17 添加 call 指令

保存修改后，重新链接生成的执行程序即可实现在 do\_phase 函数中调用 myfunc()。

```
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ hexedit phase2.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -no-pie -o linkbomb2 main.o phase2.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb2
please input you stuid : U202315653
myfunc is called. Good!
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 18 调用 myfunc



### 3) 第3关 有参数的函数调用的机器指令修改

观察反汇编代码：对于 `do_phase` 函数，寄存器 `edi` 传入参数；对于 `myfunc` 函数，由寄存器 `edi` 传入参数；由于在 `do_phase` 函数中，由起始至调用 `myfunc` 函数之间未对寄存器 `edi` 内容进行修改，因此 `myfunc` 函数中传入参数 `edi` 中的值即为 `do_phase` 函数中传入参数 `edi` 的值，即直接执行调用 `myfunc` 函数指令即可。

由于 `myfunc` 是静态函数，可以在 `phase3.o` 文件中计算 `call` 指令下一条指令的地址到 `myfunc` 的字节距离，`0-call` 指令下一条指令的地址得到一有符号负数，填充 `call` 指令位移量。`call` 指令长度为 5，`myfunc` 起始地址的偏移量为 0，因此下一条指令到 `myfunc` 的字节距离为  $0x5 + 0x33 = 0x38$ ， $0 - 0x38 = -0x38$ ，得到对应补码为 `0xffffffc8`。`call` 指令操作码为 `e8`，地址采用小端存储，因此指令为 `e8 c8 ff ff ff`。

```

● gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ objdump -d phase3.o

phase3.o:          文件格式 elf64-x86-64

Disassembly of section .text:

0000000000000000 <myfunc>:
 0:  f3 0f 1e fa          endbr64
 4:  55                   push  %rbp
 5:  48 89 e5             mov   %rsp,%rbp
 8:  48 83 ec 10          sub   $0x10,%rsp
 c:  89 7d fc             mov   %edi,-0x4(%rbp)
 f:  8b 45 fc             mov   -0x4(%rbp),%eax
12:  89 c6               mov   %eax,%esi
14:  48 8d 3d 00 00 00 00 lea   0x0(%rip),%rdi    # 1b <myfunc+0x1b>
1b:  b8 00 00 00 00      mov   $0x0,%eax
20:  e8 00 00 00 00      call  25 <myfunc+0x25>
25:  90                   nop
26:  c9                   leave
27:  c3                   ret

0000000000000028 <do_phase>:
28:  f3 0f 1e fa          endbr64
2c:  55                   push  %rbp
2d:  48 89 e5             mov   %rsp,%rbp
30:  89 7d fc             mov   %edi,-0x4(%rbp)
33:  90                   nop
34:  90                   nop
35:  90                   nop
36:  90                   nop
37:  90                   nop
38:  90                   nop
39:  90                   nop
3a:  90                   nop
3b:  90                   nop
3c:  90                   nop
3d:  90                   nop
3e:  90                   nop
3f:  90                   nop
40:  90                   nop
41:  90                   nop
42:  90                   nop
43:  90                   nop
44:  90                   nop
45:  90                   nop
46:  90                   nop
47:  90                   nop
48:  90                   nop
49:  5d                   pop   %rbp
4a:  c3                   ret

```

图 19 phase3.o 代码节

观察节表头：代码节偏移值为 0x40。

```
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ readelf -S phase3.o
There are 16 section headers, starting at offset 0x418:

节头:
[号] 名称      类型      地址      偏移量
      大小      全体大小  旗标  链接  信息  对齐
[ 0]      NULL      NULL      0000000000000000 00000000
      0000000000000000 0000000000000000 0 0 0
[ 1] .text      PROGBITS 0000000000000000 00000040
      000000000000004b AX 0 0 1
[ 2] .rela.text RELA      0000000000000000 00000310
      0000000000000030 I 13 1 8
[ 3] .data      PROGBITS 0000000000000000 0000008b
      0000000000000000 WA 0 0 1
[ 4] .bss      NOBITS   0000000000000000 0000008b
      0000000000000000 WA 0 0 1
[ 5] .rodata    PROGBITS 0000000000000000 0000008b
      0000000000000019 A 0 0 1
[ 6] .data.rel.local PROGBITS 0000000000000000 000000a8
      0000000000000008 WA 0 0 8
[ 7] .rela.data.r[...] RELA      0000000000000000 00000340
      0000000000000018 I 13 6 8
[ 8] .comment   PROGBITS 0000000000000000 000000b0
      000000000000002c MS 0 0 1
[ 9] .note.GNU-stack PROGBITS 0000000000000000 000000dc
      0000000000000000 0 0 1
[10] .note.gnu.pr[...] NOTE      0000000000000000 000000e0
      0000000000000020 A 0 0 8
[11] .eh_frame   PROGBITS 0000000000000000 00000100
      0000000000000058 A 0 0 8
[12] .rela.eh_frame RELA      0000000000000000 00000358
      0000000000000030 I 13 11 8
[13] .symtab     SYMTAB   0000000000000000 00000158
      0000000000000018 14 12 8
[14] .strtab     STRTAB   0000000000000000 000002d8
      0000000000000037 0 0 1
[15] .shstrtab   STRTAB   0000000000000000 00000388
      0000000000000089 0 0 1

Key to Flags:
W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
L (link order), O (extra OS processing required), G (group), T (TLS),
C (compressed), x (unknown), o (OS specific), E (exclude),
D (mbind), l (large), p (processor specific)
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 20 phase3.o 节表头

由上述代码节中偏移 0x33 为空得到修改  $0x40+0x33=0x73$  处，添加 call 指令。

```
00000050 45 FC 89 C6 48 8D 3D 00 00 00 00 B8 00 00 00 00 E8 00 00 00 00 90 C9 C3 F3 0F 1E FA 55 48 89 E5 89 7D FC E8 C8 FF FF FF E..H.=.....UH...}.....
```

图 21 添加 call 指令

保存修改后，重新链接生成的执行程序即可实现在 do\_phase 函数中调用 myfunc(pos)，将 do\_phase 的参数 pos 直接传递 myfunc，显示相应的信息。由关卡 1 可知参数为 8，输出结果正确。

```
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ hexedit phase3.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -no-pie -o linkbomb3 main.o phase3.o
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb3
please input you stuid : U202315653
gate 3: offset is : 8!
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 22 调用 myfunc 并传参

## 4) 第4关 有局部变量的机器指令修改

观察反汇编代码：do\_phase 函数 movabs 指令将 0x3332313232303255（小端存储）送入寄存器 rax，指令 mov %rax, -0x13(%rbp)将寄存器 rax 中数据送入栈中-0x13(%rbp)，8 个字节，小端存储；movw 指令将 0x3534 送入栈中-0xb(%rbp)，2 个字节，小端存储；movb 指令将 0x0 送入栈中-0x9(%rbp)，1 个字节。

因此栈中位置为-0x13(%rbp) ~ -0x9(%rbp)存储 0x0035343332313232303255（小端存储），即为学号字符串。

myfunc 函数通过寄存器 rdi 传入参数，其中 rdi 保存输出字符串的地址。因此实现调用 myfunc 函数从而输出学号需要将学号字符串地址保存至寄存器 rdi 中，再执行指令调用 myfunc 函数。由上述分析可知学号字符串起始地址为-0x13(%rbp)，因此执行指令 mov -0x13(%rbp), %rdi，对应机器码为 48 8d 7d ed（48：表示使用 64 位操作，确保目标寄存器%rdi 是 64 位。8d：对应 lea 指令的操作码，表示将有效地址加载到寄存器。7d：分解为 01 111 101，01 表示存在 8 位位移；111 表示目标寄存器是%rdi；101 表示基址寄存器为 %rbp。ed：8 位有符号位移-0x13 的补码）。

执行指令调用 myfunc 函数。由于 myfunc 是静态函数，可以在 phase4.o 文件中计算 call 指令下一条指令的地址到 myfunc 的字节距离，0-call 指令下一条指令的地址得到一有符号负数，填充 call 指令位移量。添加 mov 指令长度为 4，call 指令长度为 5，myfunc 起始地址的偏移量为 0，因此下一条指令到 myfunc 的字节距离， $0x4 + 0x5 + 0x61 = 0x6a$ ， $0 - 0x6a = -0x6a$ ，得到对应补码为 0xffff ff96。call 指令操作码为 e8，地址采用小端存储，因此指令为 e8 96 ff ff ff。

```

0000000000000000 <myfunc>:
0: f3 0f 1e fa      endbr64
4: 55               push    %rbp
5: 48 89 e5         mov     %rsp,%rbp
8: 48 83 ec 10      sub     $0x10,%rsp
c: 48 89 7d f8      mov     %rdi,-0x8(%rbp)
10: 48 0b 45 f8      mov     -0x8(%rbp),%rax
14: 48 89 c6         mov     %rax,%rsi
17: 48 8d 3d 00 00 00 00 lea     0x0(%rip),%rdi    # le <myfunc+0x1e>
1e: b8 00 00 00 00   mov     $0x0,%eax
23: e8 00 00 00 00   call    28 <myfunc+0x28>
28: 90               nop
29: c9               leave
2a: c3               ret

000000000000002b <do_phase>:
2b: f3 0f 1e fa      endbr64
2f: 55               push    %rbp
30: 48 89 e5         mov     %rsp,%rbp
33: 48 83 ec 30      sub     $0x30,%rsp
37: 89 7d dc         mov     %edi,-0x24(%rbp)
3a: 64 48 8b 04 25 28 00 mov     %fs:0x28,%rax
41: 00 00
43: 48 89 45 f8      mov     %rax,-0x8(%rbp)
47: 31 c0            xor     %eax,%eax
49: 48 b8 55 32 30 32 32 movabs  $0x3332313232303255,%rax
50: 31 32 33
53: 48 09 45 ed      mov     %rax,-0x13(%rbp)
57: 66 c7 45 f5 34 35 movw    $0x3534,-0xb(%rbp)
5d: c6 45 f7 00      movb    $0x0,-0x9(%rbp)
61: 90               nop
62: 90               nop
63: 90               nop
64: 90               nop
65: 90               nop
66: 90               nop
67: 90               nop
68: 90               nop
69: 90               nop
6a: 90               nop
6b: 90               nop
6c: 90               nop
6d: 90               nop
6e: 90               nop
6f: 90               nop
70: 90               nop
71: 90               nop
72: 90               nop
73: 90               nop
74: 90               nop
75: 48 8b 45 f8      mov     -0x8(%rbp),%rax
79: 64 48 33 04 25 28 00 xor     %fs:0x28,%rax
80: 00 00
82: 74 05            je      89 <do_phase+0x5e>
84: e8 00 00 00 00   call    89 <do_phase+0x5e>
89: c9               leave

```

图 23 phase4.o 代码节





## 5) 第 5 关 重定位表的修改

由实验指导可知需要将 originalclass 替换为 classname, 即修改重定位信息中的符号编号 0x000d 为 classname 对应符号 (11) 0x000b; 将 originalteacher 替换为 teachername 即修改重定位信息中的符号编号 0x000e 为 classname 对应符号 (12) 0x000c。

```

gabriell@gabriell-ThinkBook-16-G5-IRH: /media/gabriell/Data/计算机系统基础实验/实验5_链接炸弹$ readelf -s phase5.o

Symbol table '.symtab' contains 19 entries:
Num:  Value              Size Type Bind Vis      Ndx Name
 0: 0000000000000000      0 NOTYPE LOCAL DEFAULT UND
 1: 0000000000000000      0 FILE  LOCAL DEFAULT ABS phase5.c
 2: 0000000000000000      0 SECTION LOCAL DEFAULT 1 .text
 3: 0000000000000000      0 SECTION LOCAL DEFAULT 3 .data
 4: 0000000000000000      0 SECTION LOCAL DEFAULT 4 .bss
 5: 0000000000000000      0 SECTION LOCAL DEFAULT 5 .data.rel.local
 6: 0000000000000000      0 SECTION LOCAL DEFAULT 7 .rodata
 7: 0000000000000000      0 SECTION LOCAL DEFAULT 9 .note.gnu-stack
 8: 0000000000000000      0 SECTION LOCAL DEFAULT 10 .note.gnu-property
 9: 0000000000000000      0 SECTION LOCAL DEFAULT 11 .eh_frame
10: 0000000000000000      0 SECTION LOCAL DEFAULT 8 .comment
11: 0000000000000000     20 OBJECT GLOBAL DEFAULT 3 classname
12: 0000000000000020     20 OBJECT GLOBAL DEFAULT 3 teachername
13: 0000000000000040     20 OBJECT GLOBAL DEFAULT 3 originalclass
14: 0000000000000060     20 OBJECT GLOBAL DEFAULT 3 originalteacher
15: 0000000000000080      8 OBJECT GLOBAL DEFAULT 5 phase
16: 0000000000000000     87 FUNC  GLOBAL DEFAULT 1 do_phase
17: 0000000000000000      0 NOTYPE GLOBAL DEFAULT UND GLOBAL_OFFSET_TABLE
18: 0000000000000000      0 NOTYPE GLOBAL DEFAULT UND printf

gabriell@gabriell-ThinkBook-16-G5-IRH: /media/gabriell/Data/计算机系统基础实验/实验5_链接炸弹$ readelf -r phase5.o

重定位节 '.rela.text' at offset 0x3f8 contains 6 entries:
 偏移量 信息 类型 符号值 符号名称 + 加数
000000000012 000d00000002 R_X86_64_PC32 0000000000000040 originalclass - 4
000000000019 000e00000002 R_X86_64_PC32 0000000000000000 .rodata - 4
000000000023 001200000004 R_X86_64_PLT32 0000000000000000 printf - 4
00000000002a 000e00000002 R_X86_64_PC32 0000000000000060 originalteacher - 4
000000000031 000e00000002 R_X86_64_PC32 0000000000000000 .rodata + b
00000000003b 001200000004 R_X86_64_PLT32 0000000000000000 printf - 4

重定位节 '.rela.data.rel.local' at offset 0x488 contains 1 entry:
 偏移量 信息 类型 符号值 符号名称 + 加数
000000000000 001000000001 R_X86_64_64 0000000000000000 do_phase + 0

重定位节 '.rela.eh_frame' at offset 0x4a0 contains 1 entry:
 偏移量 信息 类型 符号值 符号名称 + 加数
000000000020 000200000002 R_X86_64_PC32 0000000000000000 .text + 0
  
```

图 27 查看符号编号

观察节表头: 重定位节偏移值为 0x3f8。

```

gabriell@gabriell-ThinkBook-16-G5-IRH: /media/gabriell/Data/计算机系统基础实验/实验5_链接炸弹$ readelf -S phase5.o
There are 16 section headers, starting at offset 0x548:

节头:
[ 0] 名称 类型 地址 偏移量
[ 0] 大小 全体大小 标志 链接 信息 对齐
[ 0] 0000000000000000 NULL 0000000000000000 0 0 0
[ 1] .text PROGBITS 0000000000000000 00000040
0000000000000057 0000000000000000 AX 0 0 1
[ 2] .rela.text RELA 0000000000000000 000003f8
0000000000000090 0000000000000018 I 13 1 8
[ 3] .data PROGBITS 0000000000000000 00000000
0000000000000074 0000000000000000 WA 0 0 16
[ 4] .bss NOBITS 0000000000000000 00000114
0000000000000000 0000000000000000 WA 0 0 1
[ 5] .data.rel.local PROGBITS 0000000000000000 00000118
0000000000000008 0000000000000000 WA 0 0 8
[ 6] .rela.data.r[...] RELA 0000000000000000 00000488
0000000000000018 0000000000000018 I 13 5 8
[ 7] .rodata PROGBITS 0000000000000000 00000120
0000000000000020 0000000000000000 A 0 0 1
[ 8] .comment PROGBITS 0000000000000000 00000140
000000000000002c 0000000000000001 MS 0 0 1
[ 9] .note.gnu-stack PROGBITS 0000000000000000 0000016c
0000000000000000 0000000000000000 0 0 1
[10] .note.gnu.pr[...] NOTE 0000000000000000 00000170
0000000000000020 0000000000000000 A 0 0 8
[11] .eh_frame PROGBITS 0000000000000000 00000190
0000000000000038 0000000000000000 A 0 0 8
[12] .rela.eh_frame RELA 0000000000000000 000004a0
0000000000000018 0000000000000018 I 13 11 8
[13] .symtab SYMTAB 0000000000000000 000001c8
000000000000001c8 0000000000000018 14 11 8
[14] .strtab STRTAB 0000000000000000 00000390
0000000000000064 0000000000000000 0 0 1
[15] .shstrtab STRTAB 0000000000000000 000004b8
0000000000000000 0000000000000000 0 0 1

Key to Flags:
W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
L (link order), O (extra OS processing required), G (group), T (TLS),
C (compressed), x (unknown), o (OS specific), E (exclude),
D (mbind), l (large), p (processor specific)
  
```

图 28 phase5.o 节表头

从 0x3f8 处对应修改。

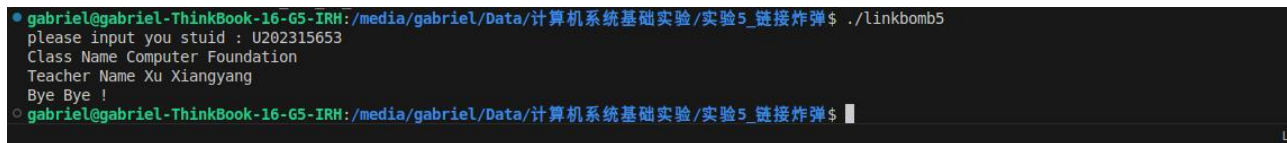
```

000003E8  42 4C 45 5F 00 70 72 69 6E 74 66 00 00 00 00 12 00 00 00 00 00 00 00 02 00 00 00 0B 00 00 00 FC FF FF FF FF FF FF FF
00000410  19 00 00 00 00 00 00 00 02 00 00 00 06 00 00 00 FC FF FF FF FF FF FF FF 23 00 00 00 00 00 00 00 04 00 00 00 12 00 00 00
00000438  FC FF FF FF FF FF FF FF 2A 00 00 00 00 00 00 00 02 00 00 00 0C 00 00 00 FC FF FF FF FF FF FF FF 31 00 00 00 00 00 00 00
  
```

图 29 修改符号编号



保存修改后，重新链接生成的执行程序即可实现显示 Class Name : Computer Foundation.  
Teacher Name : Xu Xiangyang。



```
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb5
please input you stuid : U202315653
Class Name Computer Foundation
Teacher Name Xu Xiangyang
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 30 修改后输出指定字符串

## 6) 第 6 关 强弱符号

观察反汇编代码：将地址存放在寄存器 `rax`，检测 `rax` 中值是否为 0，不为零则调用寄存器 `rdx` 中存放的地址处的函数。

图 31 phase6.o 代码节

观察得到存放在 `rax` 寄存器的值为符号 `myprint`，即检测 `myprint` 是否有内容；`myprint` 不为 0 时调用函数为地址为 `myprint` 内容的函数。因此 `myprint` 为函数指针。

图 32 查看重定位信息

观察 `linkbomb6`：验证修改前的 `myprint` 为未被赋值的函数指针。且代码中可得知无参数传递，并且无返回值。

图 33 linkbomb6 调试

查看符号编号：COM 表示 myprint 为声明的未初始化的全局变量，为弱符号。在修改 myprint 时，不同于 main.c 中的 phase 函数指针，为未定义的符号，需外部提供定义，不需要指定同名强符号进行覆盖。

```
gabriel@gabriel-ThinkBook-16-G5-IRH: /media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ readelf -s phase6.o

Symbol table '.symtab' contains 16 entries:
   Num:  Value              Size Type Bind Vis      Ndx Name
   ---:  ---              ---: ---: ---: ---: ---:  ---:
   0: 0000000000000000      0 NOTYPE LOCAL DEFAULT UND
   1: 0000000000000000      0 FILE  LOCAL DEFAULT ABS phase6.c
   2: 0000000000000000      0 SECTION LOCAL DEFAULT 1 .text
   3: 0000000000000000      0 SECTION LOCAL DEFAULT 3 .data
   4: 0000000000000000      0 SECTION LOCAL DEFAULT 4 .bss
   5: 0000000000000000      0 SECTION LOCAL DEFAULT 5 .data.rel.local
   6: 0000000000000000      0 SECTION LOCAL DEFAULT 7 .rodata
   7: 0000000000000000      0 SECTION LOCAL DEFAULT 9 .note.GNU-stack
   8: 0000000000000000      0 SECTION LOCAL DEFAULT 10 .note.gnu.property
   9: 0000000000000000      0 SECTION LOCAL DEFAULT 11 .eh_frame
  10: 0000000000000000      0 SECTION LOCAL DEFAULT 8 .comment
  11: 0000000000000000      8 OBJECT GLOBAL DEFAULT 5 phase
  12: 0000000000000000     58 FUNC  GLOBAL DEFAULT 1 do_phase
  13: 0000000000000008      8 OBJECT GLOBAL DEFAULT COM myprint
  14: 0000000000000000      0 NOTYPE GLOBAL DEFAULT UND _GLOBAL_OFFSET_TABLE_
  15: 0000000000000000      0 NOTYPE GLOBAL DEFAULT UND puts
```

图 34 查看符号编号

其中 myprint 指向函数无参数传递，因此应指定 myprint 函数指针类型为 void(\*myprint)(void)，同时初始化使其为强符号。由于链接器会优先选择强符号的定义，忽略弱符号，因此要使得 myprint 指向自定义的输出学号函数，且已经存在弱符号，必须要在 phase6\_patch.c 文件中定义 myprint 函数指针为强符号。

自定义的 printid 函数需要保持与 myprintf 指向函数类型一致，无参数与返回值，只需能够输出学号即可。

```
C main.c  C phase6_patch.c x
C phase6_patch.c > printid()
1 #include<stdio.h>
2 void printid();
3 void(*myprint)(void) = printid;
4 void printid(){
5     printf("U202315653\n");
6 }
```

图 35 phase6\_patch.c 文件

链接 phase6\_patch.o 文件生成的执行程序即可实现输出学号 U202315653。

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
gabriel@gabriel-ThinkBook-16-G5-IRH: /media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ code phase6_patch.c
gabriel@gabriel-ThinkBook-16-G5-IRH: /media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -c -g phase6_patch.c -o phase6_patch.o
gabriel@gabriel-ThinkBook-16-G5-IRH: /media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -no-pie -o linkbomb6 main.o phase6.o phase6_patch.o
gabriel@gabriel-ThinkBook-16-G5-IRH: /media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb6
please input you stuid : U202315653
U202315653
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH: /media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 36 重新链接



## 7) 第 7 关 只读数据节的修改

观察反汇编代码：将地址存放在寄存器 rdi 中。

```

gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ objdump -d phase7.o

phase7.o:          文件格式 elf64-x86-64

Disassembly of section .text:

0000000000000000 <do_phase>:
0:  f3 0f 1e fa                endbr64
4:  55                          push    %rbp
5:  48 89 e5                    mov     %rsp,%rbp
8:  48 83 ec 10                  sub     $0x10,%rsp
c:  89 7d fc                    mov     %edi,-0x4(%rbp)
f:  48 8d 3d 00 00 00 00        lea     0x0(%rip),%rdi    # 16 <do_phase+0x16>
16: e8 00 00 00 00              call    1b <do_phase+0x1b>
1b: 90                          nop
1c: c9                          leave   %rdi
1d: c3                          ret

```

图 37 phase7.o 代码节

观察得到传入地址为只读数据节，调用 puts 函数输出字符串。

```

gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ readelf -r phase7.o

重定位节 '.rela.text' at offset 0x298 contains 2 entries:
  偏移量      信息      类型      符号值      符号名称 + 加数
0000000000012 000600000000 R_X86_64_PC32 0000000000000000 .rodata - 4
0000000000017 000e00000004 R_X86_64_PLT32 0000000000000000 puts - 4

重定位节 '.rela.data.rel.local' at offset 0x2c8 contains 1 entry:
  偏移量      信息      类型      符号值      符号名称 + 加数
0000000000000 000c00000001 R_X86_64_64    0000000000000000 do_phase + 0

重定位节 '.rela.eh_frame' at offset 0x2e0 contains 1 entry:
  偏移量      信息      类型      符号值      符号名称 + 加数
0000000000020 000200000002 R_X86_64_PC32 0000000000000000 .text + 0

```

图 38 查看重定位信息

观察节表头：只读数据节偏移值为 0x2c8，大小为 0x13。

节头:						
[#]	名称	类型	地址	偏移量		
	大小	全体大小	旗标	链接	信息	对齐
[ 0]	0000000000000000	NULL	0000000000000000	0	0	0
[ 1]	.text	PROGBITS	0000000000000000	0	0	0
	000000000000001e	0000000000000000	AX	0	0	1
[ 2]	.rela.text	RELA	0000000000000000	0	0	0
	0000000000000030	0000000000000018	I	13	1	8
[ 3]	.data	PROGBITS	0000000000000000	0	0	0
	0000000000000000	0000000000000000	WA	0	0	1
[ 4]	.bss	NOBITS	0000000000000000	0	0	0
	0000000000000000	0000000000000000	WA	0	0	1
[ 5]	.data.rel.local	PROGBITS	0000000000000000	0	0	0
	0000000000000008	0000000000000000	WA	0	0	8
[ 6]	.rela.data.r[...]	RELA	0000000000000000	0	0	0
	0000000000000018	0000000000000018	I	13	5	8
[ 7]	.rodata	PROGBITS	0000000000000000	0	0	0
	0000000000000013	0000000000000000	A	0	0	1
[ 8]	.comment	PROGBITS	0000000000000000	0	0	0
	000000000000002c	0000000000000001	MS	0	0	1
[ 9]	.note.GNU-stack	PROGBITS	0000000000000000	0	0	0
	0000000000000000	0000000000000000		0	0	1
[10]	.note.gnu.pr[...]	NOTE	0000000000000000	0	0	0
	0000000000000020	0000000000000000	A	0	0	8
[11]	.eh_frame	PROGBITS	0000000000000000	0	0	0
	0000000000000038	0000000000000000	A	0	0	8
[12]	.rela.eh_frame	RELA	0000000000000000	0	0	0
	0000000000000018	0000000000000018	I	13	11	8
[13]	.symtab	SYMTAB	0000000000000000	0	0	0
	0000000000000168	0000000000000018		14	11	8
[14]	.strtab	STRTAB	0000000000000000	0	0	0
	000000000000002e	0000000000000000		0	0	1
[15]	.shstrtab	STRTAB	0000000000000000	0	0	0
	0000000000000089	0000000000000000		0	0	1

图 39 phase7.o 节表头

图 40 修改只读数据节

```
gabriel@gabriel-ThinkBook-16-Gen5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ hexedit phase7.o
gabriel@gabriel-ThinkBook-16-Gen5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ gcc -no-pie -o linkbomb7 main.o phase7.o
gabriel@gabriel-ThinkBook-16-Gen5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb7
please input you stuid : U202315653
Gate 7: U202315653
Bye Bye !
gabriel@gabriel-ThinkBook-16-Gen5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
```

图 41 输出指定学号



## 8) 最终结果

```

File Edit Selection View Go Run Terminal Help
实验5_链接炸弹
2025年_计算机系统基础_实验任务书_链接...
链接炸弹_更点提示.docx
linkbomb1
linkbomb2
linkbomb3
linkbomb4
linkbomb5
linkbomb6
linkbomb7
main.c
phase0.c
phase1.o
phase2.o
phase3.o
phase4.o
phase5.o
phase6_patch.c
phase6_patch.o
phase6.o
phase7.o

gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb1
please input you stuid : U202315653
Your ID is : U202315653
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb2
please input you stuid : U202315653
myfunc is called. Good!
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb3
please input you stuid : U202315653
gate 3: offset is : 8!
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb4
please input you stuid : U202315653
gate 4: your ID is : U202315653!
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb5
please input you stuid : U202315653
Class Name Computer Foundation
Teacher Name Xu Xiangyang
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb6
please input you stuid : U202315653
U202315653
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$ ./linkbomb7
please input you stuid : U202315653
Gate 7: U202315653
Bye Bye !
gabriel@gabriel-ThinkBook-16-G5-IRH:/media/gabriel/Data/计算机系统基础实验/实验5_链接炸弹$
  
```

图 42 实验结果

## 四、体会

### 1. 数据节与代码节的修改

数据节：直接修改数据节中的字符串需要准确定位偏移量，并确保 ASCII 码的正确性。实验中通过 hexdump 和 readelf 分析节表头，确定数据节位置后，直接替换学号对应的十六进制值，验证了数据在链接时的静态分配特性。

代码节：修改机器指令时，需熟悉汇编指令的编码方式（如 call 指令的操作码和相对位移计算）。通过反汇编分析函数地址与调用点关系，计算正确的位移补码，并注意小端存储，确保指令能正确跳转。

### 2. 重定位表的理解

重定位表记录了符号引用在链接时的修正信息。通过修改符号索引（如将 originalclass 的索引改为 classname 的索引），理解了重定位表如何解决跨模块符号引用问题。实验中需结合 readelf -r 查看重定位条目，并注意符号索引的字节序。

### 3. 强弱符号的实际应用

强弱符号的优先级在链接时至关重要。通过新增文件定义强符号 myprint，覆盖原有弱符号，实现了对函数指针的控制。这让我认识到强弱符号在库设计和模块化开发中的实用性。

### 4. 调试工具的使用

GDB 调试可执行文件时，通过反汇编和断点设置，验证了代码逻辑与寄存器传参的正确性。hexedit 和 objdump 进行二进制文件分析，帮助定位目标节和指令的物理偏移，避免盲目修改。

本次实验将理论（ELF 结构、符号解析、重定位）与实践（二进制文件修改、链接控制）紧密结合，让我对程序底层的链接过程有了更直观的认识。同时，实验中的调试与分析过程，极大提升了逆向工程能力和问题排查技巧，为后续进一步学习奠定了基础。