華中科技大學

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：图灵 2301 班

学　　号：U202311239

姓　　名：刘星佳

指导教师：王多强

2024 年 10 月 29 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

1. 第 1 关

在 phase1.o 中位置 0x0000008e 处起改为 0x55,0x32,0x30,0x32,0x33,0x31,0x31,0x32,0x33,0x39,0x00。

2. 第 2 关

在 phase2.o 中位置 0x00000062 处起改为 0xe8,0xd9,0xff,0xff,0xff

3. 第 3 关

在 phase3.o 中位置 0x00000073 处起改为 0x8b,0x7d,0xfc,0xe8,0xc5,0xff,0xff,0xff

4. 第 4 关

在 phase4.o 中位置 0x0000008b 处起改为 0x55,0x32,0x30,0x32,0x33,0x31,0x31,0x32,0x48,0x89,0x45,0xed,0x66,0xc7,0x45,0xf5,0x33,0x39,0xc6,0x45,0xf7,0x00,0x48,0x8d,0x7d,0xed,0xe8,0x96,0xff,0xff,0xff

5. 第 5 关

在 phase5.o 中位置 0x00000404 处改为 0x0b，位置 0x44c 处改为 0x0c。

6. 第 6 关

增补代码 phase6_patch.c 如下：

```
# include <stdio.h>
void printMyID(void);
void (*myprint)(void) = printMyID;
void printMyID(void) {
    puts("U202311239");
}
```

7. 第 7 关

在 phase7.o 中位置 0x00000070 处起改为 0x55,0x32,0x30,0x32,0x33,0x31,0x31,0x32,0x33,0x39,0x00。

（2） 描述修改各个文件的基本思想

1. 第 1 关

不做任何调整时运行 linkbomb1，发现程序输出了 opqrstuvwxyz0123456789，然后对二进制模块 phase1.o 反汇编，发现.data 数据段中存储了一个常量字符串 abcdefghijklmnopqrstuvwxyz0123456789，因此程序输出的结果是从这个字符串中截取的，截取位置与学号有关。所以只需要把 o 之后的字符串修改为想要的结果 U202311239 即可（字符串之后一个字节设置为 0x00 表示字符串结尾）

2. 第 2 关

反编译 linkbomb2 发现 do_phase 函数中有一大段代码空白段，因此可以在这里实现一个 call 命令调用 myfunc 函数，即 call 0x40131d，变成机器码为 e8 d9 ff ff ff（e8 后面跟着目标函数相对

下一行指令的偏移量，为-0x27），直接在二进制文件中 0x90（nop）对应的位置修改为以上机器指令即可调用 myfunc。

3. 第 3 关

这关与第 2 关很类似,传入参数只需要把参数传入 edi 寄存器即可,而先前参数存储在-0x4(%rbp)的位置，因此我们只要在 do_phase 的空代码段处加入以下汇编：

```
mov -0x4(%rbp), %edi
call <myfunc>
```

编译成二进制文件后转化成机器码，再插入到 phase3.o 空代码段对应的位置即可。

4. 第 4 关

这关我们需要现在栈帧中存储学号字符串，然后将字符串的头指针作为参数传进 myfunc 中。观察 linkbomb4 反汇编的代码，发现原先的 phase4.o 已经在-0x13(%ebp)的位置设置好了一个字符串“U202212345”，因此只需要将设置这个字符串对应的机器码部分稍作修改改为自己的学号，然后再加入以下汇编代码：

```
lea -0x13(%rbp), %rdi
call <myfunc>
```

用 gdb 和 objdump 得到机器码后修改 phase4.o 即可。

5. 第 5 关

首先通过 readelf -r 查看重定位节的内容：

```
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  000600000002 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  000600000002 R_X86_64_PC32     0000000000000000 .rodata + b
00000000003b  001200000004 R_X86_64_PLT32    0000000000000000 printf - 4
```

发现 printf 对应的偏移量在汇编的 do_phase 中正好是两个 call,因此输出是在 do_phase 中进行的，并且可以发现 printf 传入的参数对应的符号为 originalclass 和 originalteacher。这时候用 readelf -s 看一眼符号表：

```
  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
```

根据偏移量对照汇编中.data 节内容，发现 originalclass 是 c programming，而 classname 和 teachername 是 Computer Foundation 和 Xu Xiangyang，意味着我们要修改重定位表使得输出的内容应当定位到符号 classname 和 teachername 处。

在重定位节中，可以发现 r_info（信息）字段的高 16 位为 r_sym，表示对应的符号在符号表中的编号；低 32 位为 r_type 表示重定位类型。我们应该改变 r_sym，将偏移量 0x12 对应的 r_sym 从 000d 改为 000b（对应符号表 11，classname），偏移量为 0x2a 对应的 r_sym 从 000e 改为 000c（对应符号

表 12，teachername）即可。

要着手修改，还需要知道重定位节在.o 文件的位置（偏移量），使用 readelf -S 命令查看节头：

```
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             0000000000000000  00000000
       0000000000000000  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
```

在 0x3f8 处修改。

6. 第 6 关

首先查看重定位表：

```
重定位节 '.rela.text' at offset 0x2e8 contains 4 entries:
  偏移量          信息           类型           符号值        符号名称 + 加数
000000000012  000d00000002 R_X86_64_PC32     0000000000000008 myprint - 4
00000000001e  000d00000002 R_X86_64_PC32     0000000000000008 myprint - 4
00000000002e  000600000002 R_X86_64_PC32     0000000000000000 .rodata - 4
000000000033  000f00000004 R_X86_64_PLT32    0000000000000000 puts - 4
```

发现这里有一个陌生的符号 myprint，结合题目要求可以认为这是一个我们需要在 phase6_patch.c 中自己实现的函数。查看反汇编的结果，发现 myprint 作为一个指针（函数指针）在寄存器间传递，最终用 call 调用。同时，在调用这个函数之前也没有对往 rdi 寄存器中进行赋值，因此这个函数没有参数，调用这个函数之后也没有用到 eax/rax 寄存器的值，因此这个函数也没有返回值。所以，我们只需要先实现一个自己的函数 void printMyID(void)，作用就是输出自己的学号，然后再赋值给 extern 函数指针 myprint 即可。

使用 gcc -c phase6_patch.c 得到可重定位目标文件 phase6_patch.o 并和 phase6.o 一起链接到 main.c 上。

7. 第 7 关

仍然先看重定位表：

```
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
000000000012  000600000002 R_X86_64_PC32     0000000000000000 .rodata - 4
000000000017  000e00000004 R_X86_64_PLT32    0000000000000000 puts - 4
```

偏移量 12 对应的是传入给 rdi 寄存器即下一步调用的函数的参数，而下一步调用的函数位于偏移量 17，总而言之就是用 puts 输出了位于.rodata 节的数据。查看 phase7.o 的二进制文件发现.rodata 节是一个字符串“Gate 7: U202212345”，因此将这里的学号改为自己的学号即可。

四、体会

通过这次《计算机系统基础》课程的实验，我对 ELF 文件与程序链接的理论知识有了更深刻的理解。实验不仅加深了我对可重定位目标文件格式、目标文件的生成以及链接过程的认识，而且通过实际操作，我学会了如何使用各种工具（如 GCC、GDB、readelf 等）来分析和修改二进制文件。

第 1 关让我学会了如何修改数据节来改变程序的输出，这不仅需要理解数据节的作用，还需要精确地定位到需要修改的位置。第 2 关和第 3 关则让我掌握了如何在代码节中插入机器指令，这对于理解程序的执行流程和函数调用机制非常有帮助。第 4 关难点在于理解局部变量和函数参数的传递，这让我对栈帧和函数调用约定有了更深入的认识。第 5 关花了我最长的时间，因为它涉及到重定位表的修改。这不仅需要理解重定位表的结构和作用，还需要精确地定位到需要修改的重定位条目。通过这个关卡，我对 ELF 文件的结构和链接过程的理解更加深入。第 6 关和第 7 关我学会了如何处理强弱符号和只读数据节的修改。

这次实验不仅提高了我的技术能力，也锻炼了我的问题解决能力。我学会了如何分析问题、查找资料、动手实践，并最终解决问题。这些经验对我的学习和未来都是非常宝贵的。我会将这些知识应用到更复杂的项目中，继续深化我的理解和技能。