实验6

班级：
姓名：
学号：

实验内容：

1 链接与 ELF 实验的内容、方法和基本步骤；
2 程序链接的作用与过程、ELF 文件格式组成等知识的回顾与应用。

实验目标：

1 加深对程序链接中符号解析、重定位等基本概念、位置无关代码和 ELF 文件的基本组成等方面知识的理解和掌握；
2 掌握计算机系统思维，理解高级语言中数据、运算、过程调用和 I/O 操作等在计算机系统中的实现方法，将程序设计、汇编语言、系统结构、操作系统、编译链接中的重要概念贯穿起来。能够对计算机系统复杂工程问题制定解决方案
3 掌握各种开源的编译调试工具，能够对分析优化程序设计，提高在代码调试、性能提升、软件移植和鲁棒性等方面的能力。

实验任务：

1 学 习 MOOC 内 容
https://www.icourse163.org/learn/NJU-1449521162
第七周 程序的链接
第 1 讲 链接与 ELF 实验：概述
第 2 讲 链接与 ELF 实验：静态数据与 ELF 数据节
第 3 讲 链接与 ELF 实验：指令与 ELF 代码节及课后实验

2 完成实验
详见链接与 ELF 实验文档
2.1 静态数据对象与 ELF 数据节 phase1
2.1.1 程序代码
首先对程序进行链接并执行，查看结果：

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ gcc -no-pie -o lb1 main.o phase1.o
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ ./lb1
ddURHzFnm2mcxbehqVcVufpd68LdEePs        lYPCnZfPLLbMLzV3iM1A97QVLg7j8zcmDlD0clCtKV0kgLRshaBQ3kCaGG YMbr
9ELE31xt2fau4zX7bEMCVf  qXOdnQ igVJcDsac1d9N7kSla5VLXAKDtjxAoNjW2tonwDzyASqLn5JKSf32EqapXP83B03NmDqUx
```

然后反汇编phase1.o，查看结果：

```
phase1.o:     文件格式 elf32-i386


Disassembly of section .text:

00000000 <do_phase>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 ec 08                sub    $0x8,%esp
   6:   b8 9a 00 00 00          mov    $0x9a,%eax
   b:   83 ec 0c                sub    $0xc,%esp
   e:   50                      push   %eax
   f:   e8 fc ff ff ff          call   10 <do_phase+0x10>
  14:   83 c4 10                add    $0x10,%esp
  17:   90                      nop
  18:   c9                      leave
  19:   c3                      ret
```

然后用readelf -r 查看phase.o对应的信息：

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ readelf -r phase1.o

重定位节 '.rel.text' 位于偏移量 0x354 含有 2 个条目:
 偏移量     信息     类型              符号值      符号名称
00000007  00000301 R_386_32          00000000   .data
00000010  00000e02 R_386_PC32        00000000   puts

重定位节 '.rel.data' 位于偏移量 0x364 含有 2 个条目:
 偏移量     信息     类型              符号值      符号名称
00000068  00000601 R_386_32          00000000   .rodata
00000160  00000d01 R_386_32          00000000   do_phase

重定位节 '.rel.eh_frame' 位于偏移量 0x374 含有 1 个条目:
 偏移量     信息     类型              符号值      符号名称
00000020  00000202 R_386_PC32        00000000   .text
```

利用readelf -x .data phase1.o获得data节的内容：

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ readelf -x .data phase1.o

“.data”节的十六进制输出:
 NOTE: This section has relocations against it, but these have NOT been applied to this dump.
  0x00000000 eb7376f3 ee5e59e3 905c9348 6117edf7 .sv..^Y..\.Ha...
  0x00000010 d4d0a234 a37763f3 de71ac68 361e3be6 ...4.wc..q.h6.;.
  0x00000020 08e3cc1a 7e127d65 1df241c7 5ddab329 ....~.}e..A.]..)
  0x00000030 4c0db066 c017e14f 526ee47a 030b14fa L..f...ORn.z....
  0x00000040 b582492e 432e303f 9b03f839 5ef47119 ..I.C.0?...9^.q.
  0x00000050 87f3e290 9bf69d17 012e4cdf 214fb0fa ..........L.!O..
  0x00000060 32fe4a4f 2304e054 00000000 00000000 2.JO#..T........
  0x00000070 00000000 00000000 00000000 00000000 ................
  0x00000080 4f663657 70705551 30506579 356e5836 Of6WppUQ0Pey5nX6
  0x00000090 47552051 41356c78 76616464 5552487a GU QA5lxvaddURHz
  0x000000a0 466e6d32 6d637862 65687156 63567566 Fnm2mcxbehqVcVuf
  0x000000b0 70643638 4c644565 5073096c 5950436e pd68LdEePs.lYPCn
  0x000000c0 5a66504c 4c624d4c 7a563369 4d314139 ZfPLLbMLzV3iM1A9
  0x000000d0 3751564c 67376a38 7a636d44 6c443063 7QVLg7j8zcmDlD0c
  0x000000e0 6c43744b 56306b67 4c527368 61425133 lCtKV0kgLRshaBQ3
  0x000000f0 6b436147 4720594d 62723945 4c453331 kCaGG YMbr9ELE31
  0x00000100 78743266 6175347a 58376245 4d435666 xt2fau4zX7bEMCVf
  0x00000110 0971584f 646e5120 6967564a 63447361 .qXOdnQ igVJcDsa
  0x00000120 63316439 4e376b53 6c613556 4c58414b c1d9N7kSla5VLXAK
  0x00000130 44746a78 416f4e6a 5732746f 6e77447a DtjxAoNjW2tonwDz
  0x00000140 79415371 4c6e354a 4b536633 32457161 yASqLn5JKSf32Eqa
  0x00000150 70585038 33423033 4e6d4471 55780008 pXP83B03NmDqUx..
  0x00000160 00000000                             ....
```

发现输出字符串对应的地址确实是data节中的0x9a。然后利用readeif -S phase1.o 查看data节的偏移量：

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ readelf -S phase1.o
共有 14 个节头，从偏移量 0x3e0 开始:

节头:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .text             PROGBITS        00000000 000034 00001a 00  AX  0   0  1
  [ 2] .rel.text         REL             00000000 000354 000010 08   I 11   1  4
  [ 3] .data             PROGBITS        00000000 000060 000164 00  WA  0   0 32
  [ 4] .rel.data         REL             00000000 000364 000010 08   I 11   3  4
  [ 5] .bss              NOBITS          00000000 0001c4 000000 00  WA  0   0  1
  [ 6] .rodata           PROGBITS        00000000 0001c4 000002 00   A  0   0  1
  [ 7] .comment          PROGBITS        00000000 0001c6 00001d 01  MS  0   0  1
  [ 8] .note.GNU-stack   PROGBITS        00000000 0001e3 000000 00      0   0  1
  [ 9] .eh_frame         PROGBITS        00000000 0001e4 000038 00   A  0   0  4
  [10] .rel.eh_frame     REL             00000000 000374 000008 08   I 11   9  4
  [11] .symtab           SYMTAB          00000000 00021c 000100 10     12  12  4
  [12] .strtab           STRTAB          00000000 00031c 000038 00      0   0  1
  [13] .shstrtab         STRTAB          00000000 00037c 000063 00      0   0  1
```

发现.data节的偏移量为0x60，所以输出字符串在文件中的偏移量为0x60 + 0x9a = 0xfa。

然后利用hexedit工具打开phase1.o，修改偏移量为0xfa的字节处：

```
00000000   7F 45 4C 46   01 01 01 00   00 00 00 00   00 00 00 00   01 00 03 00   .ELF............
00000014   01 00 00 00   00 00 00 00   00 00 00 00   E0 03 00 00   00 00 00 00   ................
00000028   34 00 00 00   00 00 28 00   0E 00 0D 00   55 89 E5 83   EC 08 B8 9A   4.....(.....U.......
0000003C   00 00 00 83   EC 0C 50 E8   FC FF FF FF   83 C4 10 90   C9 C3 00 00   ......P.............
00000050   00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00   EB 73 76 F3   .................sv.
00000064   EE 5E 59 E3   90 5C 93 48   61 17 ED F7   D4 D0 A2 34   A3 77 63 F3   .^Y..\.Ha......4.wc.
00000078   DE 71 AC 68   36 1E 3B E6   08 E3 CC 1A   7E 12 7D 65   1D F2 41 C7   .q.h6.;.....~.}e..A.
0000008C   5D DA B3 29   4C 0D B0 66   C0 17 E1 4F   52 6E E4 7A   03 0B 14 FA   ]..)L..f...ORn.z....
000000A0   B5 82 49 2E   43 2E 30 3F   9B 03 F8 39   5E F4 71 19   87 F3 E2 90   ..I.C.0?...9^.q.....
000000B4   9B F6 9D 17   01 2E 4C DF   21 4F B0 FA   32 FE 4A 4F   23 04 E0 54   ......L.!O..2.JO#..T
000000C8   00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00   ................
000000DC   00 00 00 00   4F 66 36 57   70 70 55 51   30 50 65 79   35 6E 58 36   ....Of6WppUQ0Pey5nX6
000000F0   47 55 20 51   41 35 6C 78   76 61 32 30   32 31 31 30   30 33 36 38   GU QA5lxva2021100368
00000104   36 00 78 62   65 68 71 56   63 56 75 66   70 64 36 38   4C 64 45 65   6.xbehqVcVufpd68LdEe
00000118   50 73 09 6C   59 50 43 6E   5A 66 50 4C   4C 62 4D 4C   7A 56 33 69   Ps.lYPCnZfPLLbMLzV3i
0000012C   4D 31 41 39   37 51 56 4C   67 37 6A 38   7A 63 6D 44   6C 44 30 63   M1A97QVLg7j8zcmDlD0c
00000140   6C 43 74 4B   56 30 6B 67   4C 52 73 68   61 42 51 33   6B 43 61 47   lCtKV0kgLRshaBQ3kCaG
00000154   47 20 59 4D   62 72 39 45   4C 45 33 31   78 74 32 66   61 75 34 7A   G YMbr9ELE31xt2fau4z
00000168   58 37 62 45   4D 43 56 66   09 71 58 4F   64 6E 51 20   69 67 56 4A   X7bEMCVf.qXOdnQ igVJ
0000017C   63 44 73 61   63 31 64 39   4E 37 6B 53   6C 61 35 56   4C 58 41 4B   cDsac1d9N7kSla5VLXAK
00000190   44 74 6A 78   41 6F 4E 6A   57 32 74 6F   6E 77 44 7A   79 41 53 71   DtjxAoNjW2tonwDzyASq
000001A4   4C 6E 35 4A   4B 53 66 33   32 45 71 61   70 58 50 38   33 42 30 33   Ln5JKSf32EqapXP83B03
000001B8   4E 6D 44 71   55 78 00 08   00 00 00 00   31 00 00 47   43 43 3A 20   NmDqUx......1..GCC:
000001CC   28 44 65 62   69 61 6E 20   38 2E 33 2E   30 2D 36 29   20 38 2E 33   (Debian 8.3.0-6) 8.3
000001E0   2E 30 00 00   14 00 00 00   00 00 00 00   01 7A 52 00   01 7C 08 01   .0...........zR..|..
000001F4   1B 0C 04 04   88 01 00 00   1C 00 00 00   1C 00 00 00   00 00 00 00   ................
00000208   1A 00 00 00   00 41 0E 08   85 02 42 0D   05 56 C5 0C   04 04 00 00   .....A....B..V......
0000021C   00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00   01 00 00 00   ................
00000230   00 00 00 00   00 00 00 00   04 00 F1 FF   00 00 00 00   00 00 00 00   ................
-**  phase1.o       --0x106/0x610-----------------------------------------------------------------
```

2.1.2 结果分析与讨论

重新链接并执行，成功输出学号：

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ gcc -no-pie -o lb1 main.o phase1.o
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ ./lb1
20211003686
```

2.2 指令与 ELF 代码节 phase2

2.2.1 程序代码

链接phase2.o模块和main.o模块，然后运行查看结果：

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ gcc -no-pie -o lb2 main.o phase2.o
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ ./lb2
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$
```

可知，phase2默认不输出任何信息。

利用objdump对phase2进行反汇编，并查看代码：

```
0804846b <main>:
 804846b:       8d 4c 24 04             lea    0x4(%esp),%ecx
 804846f:       83 e4 f0                and    $0xfffffff0,%esp
 8048472:       ff 71 fc                pushl  -0x4(%ecx)
 8048475:       55                      push   %ebp
 8048476:       89 e5                   mov    %esp,%ebp
 8048478:       51                      push   %ecx
 8048479:       83 ec 04                sub    $0x4,%esp
 804847c:       a1 28 a0 04 08          mov    0x804a028,%eax
 8048481:       85 c0                   test   %eax,%eax
 8048483:       74 09                   je     804848e <main+0x23>
 8048485:       a1 28 a0 04 08          mov    0x804a028,%eax
 804848a:       ff d0                   call   *%eax
 804848c:       eb 10                   jmp    804849e <main+0x33>
 804848e:       83 ec 0c                sub    $0xc,%esp
 8048491:       68 30 87 04 08          push   $0x8048730
 8048496:       e8 95 fe ff ff          call   8048330 <puts@plt>
 804849b:       83 c4 10                add    $0x10,%esp
 804849e:       b8 00 00 00 00          mov    $0x0,%eax
 80484a3:       8b 4d fc                mov    -0x4(%ebp),%ecx
 80484a6:       c9                      leave
 80484a7:       8d 61 fc                lea    -0x4(%ecx),%esp
 80484aa:       c3                      ret
```

然后利用readelf查看phase2中出现的重定位记录：

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ readelf -r phase2.o

重定位节 '.rel.text' 位于偏移量 0x308 含有 4 个条目:
 偏移量     信息    类型              符号值      符号名称
00000035  00000c02 R_386_PC32        00000000   strlen
0000006b  00000501 R_386_32          00000000   .rodata
00000073  00000d02 R_386_PC32        00000000   strcmp
00000085  00000e02 R_386_PC32        00000000   puts

重定位节 '.rel.data' 位于偏移量 0x328 含有 2 个条目:
 偏移量     信息    类型              符号值      符号名称
00000000  00000501 R_386_32          00000000   .rodata
00000004  00000f01 R_386_32          00000091   do_phase

重定位节 '.rel.eh_frame' 位于偏移量 0x338 含有 3 个条目:
 偏移量     信息    类型              符号值      符号名称
00000020  00000202 R_386_PC32        00000000   .text
00000040  00000202 R_386_PC32        00000000   .text
00000060  00000202 R_386_PC32        00000000   .text
```

在phase2.o代码节中找到包含有类似puts输出函数调用的函数

```
 81:   ff 75 0c                   pushl  0xc(%ebp)
 84:   e8 fc ff ff ff             call   85 <kfSvKnbh+0x24>
```

查找重定位表，发现其对应puts函数。所以调用puts语句的函数即为输出字符串的函数，然后就可以在phase2模块的do_phase函数中调用kfSvKnbh函数以实现字符串的输出。

利用readelf查看phase2的符号表：

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ readelf -s phase2.o

Symbol table '.symtab' contains 17 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 00000000     0 FILE    LOCAL  DEFAULT  ABS phase2.c
     2: 00000000     0 SECTION LOCAL  DEFAULT    1
     3: 00000000     0 SECTION LOCAL  DEFAULT    3
     4: 00000000     0 SECTION LOCAL  DEFAULT    5
     5: 00000000     0 SECTION LOCAL  DEFAULT    6
     6: 00000061    48 FUNC    LOCAL  DEFAULT    1 kfSvKnbh
     7: 00000000     0 SECTION LOCAL  DEFAULT    8
     8: 00000000     0 SECTION LOCAL  DEFAULT    9
     9: 00000000     0 SECTION LOCAL  DEFAULT    7
    10: 00000000     4 OBJECT  GLOBAL DEFAULT    3 phase_id
    11: 00000000    97 FUNC    GLOBAL DEFAULT    1 OOtZxJJdNH
    12: 00000000     0 NOTYPE  GLOBAL DEFAULT  UND strlen
    13: 00000000     0 NOTYPE  GLOBAL DEFAULT  UND strcmp
    14: 00000000     0 NOTYPE  GLOBAL DEFAULT  UND puts
    15: 00000091    70 FUNC    GLOBAL DEFAULT    1 do_phase
    16: 00000004     4 OBJECT  GLOBAL DEFAULT    3 phase
```

可知puts函数在.text节中的偏移量为0x61，对应编号为1的节，大小为48。
然后利用readelf查看节头表：

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ readelf -S phase2.o
共有 14 个节头，从偏移量 0x3b4 开始：

节头：
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .text             PROGBITS        00000000 000034 0000d7 00  AX  0   0  1
  [ 2] .rel.text         REL             00000000 000308 000020 08   I 11   1  4
  [ 3] .data             PROGBITS        00000000 00010c 000008 00  WA  0   0  4
  [ 4] .rel.data         REL             00000000 000328 000010 08   I 11   3  4
  [ 5] .bss              NOBITS          00000000 000114 000000 00  WA  0   0  1
  [ 6] .rodata           PROGBITS        00000000 000114 00000a 00   A  0   0  1
  [ 7] .comment          PROGBITS        00000000 00011e 00001d 01  MS  0   0  1
  [ 8] .note.GNU-stack   PROGBITS        00000000 00013b 000000 00      0   0  1
  [ 9] .eh_frame         PROGBITS        00000000 00013c 000078 00   A  0   0  4
  [10] .rel.eh_frame     REL             00000000 000338 000018 08   I 11   9  4
  [11] .symtab           SYMTAB          00000000 0001b4 000110 10     12  10  4
  [12] .strtab           STRTAB          00000000 0002c4 000043 00      0   0  1
  [13] .shstrtab         STRTAB          00000000 000350 000063 00      0   0  1
```

可知编号为1的节是text节，所以目标函数位于text节中偏移量为0x61的位置上。
查看kfSvKnbh函数的汇编代码：

```
00000061 <kfSvKnbh>:
  61:   55                      push   %ebp
  62:   89 e5                   mov    %esp,%ebp
  64:   83 ec 08                sub    $0x8,%esp
  67:   83 ec 08                sub    $0x8,%esp
  6a:   68 02 00 00 00          push   $0x2
  6f:   ff 75 08                pushl  0x8(%ebp)
  72:   e8 fc ff ff ff          call   73 <kfSvKnbh+0x12>
  77:   83 c4 10                add    $0x10,%esp
  7a:   85 c0                   test   %eax,%eax
  7c:   75 10                   jne    8e <kfSvKnbh+0x2d>
  7e:   83 ec 0c                sub    $0xc,%esp
  81:   ff 75 0c                pushl  0xc(%ebp)
  84:   e8 fc ff ff ff          call   85 <kfSvKnbh+0x24>
  89:   83 c4 10                add    $0x10,%esp
  8c:   eb 01                   jmp    8f <kfSvKnbh+0x2e>
  8e:   90                      nop
  8f:   c9                      leave
  90:   c3                      ret
```

查看第一个call语句的重定位记录可知其调用的是strcmp函数，strcmp函数比较传递给它的两个字符串参数的内容是否相同，如果相同的话将返回0，如不同则返回非0的一个数。
当存放于EAX寄存器中的、strcmp函数的返回值等于0，即两个字符串相同时，将执行jne跳转指令后的参数压栈指令和调用puts函数的另一call指令。这个call对应的就是puts输出函数，被输出的、传递给puts函数的参数字符串地址就是传递给kfSvKnbh函数的第二个参数。
进一步使用readelf工具输出模块中 .rodata节的内容:

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ readelf -x .rodata phase2.o

“.rodata”节的十六进制输出：
  0x00000000 32004d53 4c756c65 5800             2.MSLuleX.
```

其内容为“MSLuleX.”
内置字符串地址＝.rodata节起始地址＋引用处的初始值0x2。
然后在栈中构造与内置字符串和学号字符串内容相同的局部字符串变量，并将其地址作为实参压入栈中 。构造在do_phase函数中，调用kfSvKnbh函数的机器指令代码。

```
sub     $0x28,%esp
movl    $0x566f6c49,-0x10(%ebp)
movl    $0x536345,-0xc(%ebp)
movl    $0x31323032,-0x1a(%ebp)
movl    $0x33303031,-0x16(%ebp)
movw    $0x363836,-0x12(%ebp)
sub     $0x8,%esp
lea     -0x1a(%ebp),%eax
push    %eax
lea     -0x10(%ebp),%eax
push    %eax
call    0x00
add     $0x10,%esp
nop
leave
ret
```

然后利用as工具进行汇编，再利用objdump进行反汇编并查看机器码：

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ as -o a2_main.o a2_main.s
a2_main.s: Assembler messages:
a2_main.s:6: 警告:  0000000000363836 shortened to 0000000000003836
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ objdump -d a2_main.o

a2_main.o:     文件格式 elf32-i386


Disassembly of section .text:

00000000 <.text>:
   0:   83 ec 28                sub    $0x28,%esp
   3:   c7 45 f0 49 6c 6f 56    movl   $0x566f6c49,-0x10(%ebp)
   a:   c7 45 f4 45 63 53 00    movl   $0x536345,-0xc(%ebp)
  11:   c7 45 e6 32 30 32 31    movl   $0x31323032,-0x1a(%ebp)
  18:   c7 45 ea 31 30 30 33    movl   $0x33303031,-0x16(%ebp)
  1f:   66 c7 45 ee 36 38       movw   $0x3836,-0x12(%ebp)
  25:   83 ec 08                sub    $0x8,%esp
  28:   8d 45 e6                lea    -0x1a(%ebp),%eax
  2b:   50                      push   %eax
  2c:   8d 45 f0                lea    -0x10(%ebp),%eax
  2f:   50                      push   %eax
  30:   e8 fc ff ff ff          call   0x31
  35:   83 c4 10                add    $0x10,%esp
  38:   90                      nop
  39:   c9                      leave
  3a:   c3                      ret
```

使用上述构造的调用输出函数的指令代码，替换do_phase()函数体中的nop指令,以实现期望输出。

使用readelf工具及其“-S”命令行选项获得节头表，从中获得.text在phase2.o中偏移量为 0x34，进一步得到函数中插入指令代码的位置为：0x34 + 0xcf = 0xc8 （0xcf是do_phase函数中第一个被替换的nop指令的偏移量）。

使用hexedit将phase2.o文件中上述位置0xc8起始的字节内容（原为nop指令代码0x90）替换为调用指令代码序列：

```
00000000   7F 45 4C 46  01 01 01 00  00 00 00 00  00 00 00 00  01 00 03 00  .ELF............
00000014   01 00 00 00  00 00 00 00  00 00 00 00  B4 03 00 00  00 00 00 00  ................
00000028   34 00 00 00  00 00 28 00  0E 00 0D 00  55 89 E5 83  EC 28 C7 45  4.....(.....U....(.E
0000003C   DF 32 68 56  44 C7 45 E3  69 33 71 74  C7 45 E7 64  69 4B 79 C7  .2hVD.E.i3qt.E.diKy.
00000050   45 EB 4D 54  6F 77 C7 45  EF 53 48 37  09 C6 45 F3  00 83 EC 0C  E.MTow.E.SH7..E.....
00000064   8D 45 DF 50  E8 FC FF FF  FF 83 C4 10  89 45 F4 83  7D 08 00 78  .E.P.........E..}..x
00000078   15 8B 45 08  3B 45 F4 7D  0D 8D 55 DF  8B 45 08 01  D0 0F B6 00  ..E.;E.}..U..E......
0000008C   EB 05 B8 00  00 00 00 C9  C3 55 89 E5  83 EC 08 83  EC 08 68 02  .........U........h.
000000A0   00 00 00 FF  75 08 E8 FC  FF FF FF 83  C4 10 85 C0  75 10 83 EC  ....u...........u...
000000B4   0C FF 75 0C  E8 FC FF FF  FF 83 C4 10  EB 01 90 C9  C3 55 89 E5  ..u.............U..
000000C8   83 EC 28 C7  45 F0 49 6C  6F 56 C7 45  F4 45 63 53  00 C7 45 E6  ..(.E.IloV.E.EcS..E.
000000DC   32 30 32 31  C7 45 EA 31  30 30 33 66  C7 45 EE 36  38 83 EC 08  2021.E.1003f.E.68...
000000F0   8D 45 E6 50  8D 45 F0 50  E8 FC FF FF  FF 83 C4 10  90 C9 C3 90  .E.P.E.P........
00000104   90 90 90 90  90 5D C3 00  00 00 00 00  00 00 00 00  32 00 4D 53  .....]..........2.MS
00000118   4C 75 6C 65  58 00 00 47  43 43 3A 20  28 44 65 62  69 61 6E 20  LuleX..GCC: (Debian
0000012C   38 2E 33 2E  30 2D 36 29  20 38 2E 33  2E 30 00 00  14 00 00 00  8.3.0-6) 8.3.0......
00000140   00 00 00 00  01 7A 52 00  01 7C 08 01  1B 0C 04 04  88 01 00 00  .....zR..|..........
00000154   1C 00 00 00  1C 00 00 00  00 00 00 00  61 00 00 00  00 41 0E 08  ............a....A..
00000168   85 02 42 0D  05 02 5D C5  0C 04 04 00  1C 00 00 00  3C 00 00 00  ..B...]..........<...
0000017C   61 00 00 00  30 00 00 00  00 41 0E 08  85 02 42 0D  05 6C C5 0C  a...0....A....B..l..
00000190   04 04 00 00  1C 00 00 00  5C 00 00 00  91 00 00 00  46 00 00 00  ........\.......F...
000001A4   00 41 0E 08  85 02 42 0D  05 02 42 C5  0C 04 04 00  00 00 00 00  .A....B...B.........
000001B8   00 00 00 00  00 00 00 00  00 00 00 00  01 00 00 00  00 00 00 00  ....................
000001CC   00 00 00 00  04 00 F1 FF  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000001E0   03 00 01 00  00 00 00 00  00 00 00 00  00 00 00 00  03 00 03 00  ....................
000001F4   00 00 00 00  00 00 00 00  00 00 00 00  03 00 05 00  00 00 00 00  ....................
00000208   00 00 00 00  00 00 00 00  03 00 06 00  0A 00 00 00  61 00 00 00  ............a...
0000021C   30 00 00 00  02 00 01 00  00 00 00 00  00 00 00 00  00 00 00 00  0...............
00000230   03 00 08 00  00 00 00 00  00 00 00 00  00 00 00 00  03 00 09 00  ................
```

2.2.3 结果分析与讨论

重新链接并执行，成功输出学号：

```
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ gcc -no-pie -o linkbomb main.o phase1.o
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ ./lingkbomb
bash: ./lingkbomb: 没有那个文件或目录
sanfenbai@ubuntu:~/Desktop/计算机系统/课程设计/链接$ ./linkbomb
20211003686
```