哈爾濱工業大學

实验报告

实 验（五）

题　　目　LinkLab

链接

专　　业　计算机科学与技

学　　号　L170300901

班　　级　170300901

学　　生　卢兑玑

指 导 教 师　史先俊

实 验 地 点　G712

实 验 日 期

计算机科学与技术学院

目 录

第 1 章 实验基本信息

1.1 实验目的

理解链接的作用与工作步骤
掌握 ELF 结构与符号解析与重定位的工作过程
熟练使用 Linux 工具完成 ELF 分析与修改

1.2 实验环境与工具

1.2.1 硬件环境

X64 CPU；2GHz；2G RAM；256GHD Disk 以上

1.2.2 软件环境

Windows7 64 位以上；VirtualBox/Vmware 11 以上；Ubuntu 16.04 LTS 64 位/优麒麟 64 位；

1.2.3 开发工具

Visual Studio 2010 64 位以上；GDB/OBJDUMP；DDD/EDB 等

1.3 实验预习

上实验课前，必须认真预习实验指导书（PPT 或 PDF）
了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。

请按顺序写出 ELF 格式的可执行目标文件的各类信息

请按照内存地址从低到高的顺序，写出 Linux 下 X64 内存映像。

请运行“LinkAddress -u 学号 姓名” 按地址循序写出各符号的地址、空间。并按照 Linux 下 X64 内存映像标出其所属各区。

请按顺序写出 LinkAddress 从开始执行到 main 前/后执行的子程序的名字。(gcc 与 objdump/GDB/EDB)

第 2 章 实验预习

2.1 请按顺序写出 ELF 格式的可执行目标文件的各类信息（5 分）

ELF 头
段头部表
.init
.text
.rodata
.data
.bss
.symtab
.debug
.line
.strtab
节头部表

2.2 请按照内存地址从低到高的顺序，写出 Linux 下 X64 内存映像。（5 分）

内核内存
用户栈 （运行时 创建）
（栈-向下） . .

. （映射区域-向上）
共享库的内存映射区域
. . . （堆-向上）
运行时堆 （由 malloc 创建）

2.3 请运行“LinkAddress -u 学号 姓名” 按地址循序写出各符号的地址、空间。并按照 Linux 下 X64 内存映像标出其所属各区。

（5 分）

所属	符号、地址、空间（从小到大）
0（NULL）	p5 (nil) 0
只读代码段（.init，.text，.rodata）	show_pointer 0x55890bd0681a 94047097088026 useless 0x55890bd0684d 94047097088077 main 0x55890bd06858 94047097088088
读/写段	global 0x55890bf0802c 94047099191340 huge array

（.data，.bss）	0x55890bf08040 94047099191360 big array 0x55894bf08040 94048172933184 p2 0x55894ead4670 94048218859120
运行时堆	p1 0x7f442c23c010 139930775044112 p3 0x7f443c819010 139931049627664 p4 0x7f43ec23b010 139929701298192
共享库的内存 映射区域	exit 0x7f443c280120 139931043758368 printf 0x7f443c2a1e80 139931043896960 malloc 0x7f443c2d4070 139931044102256 free 0x7f443c2d4950 139931044104528
用户栈 （运行时创 建）	. . . argc 0x7ffebe3ea1ac 140732090196396 local 0x7ffebe3ea1b0 140732090196400 argv 0x7ffebe3ea2d8 140732090196696 argv[0] 7ffebe3ec231 argv[1] 7ffebe3ec23f argv[2] 7ffebe3ec242 argv[3] 7ffebe3ec24d argv[0] 0x7ffebe3ec231 140732090204721 ./linkaddress argv[1] 0x7ffebe3ec23f 140732090204735

	-u argv[2] 0x7ffebe3ec242 140732090204738 L170300901 argv[3] 0x7ffebe3ec24d 140732090204749 卢兑玳

2.4 请按顺序写出 LinkAddress 从开始执行到 main 前/后执行的子程序的名字。（gcc 与 objdump/GDB/EDB）（5 分）

时间段	程序
Main 函数执行前	Ld-2.27.so!_dl_start Ld-2.27.so!_dl_init Libc-2.27.so!_cxa_atexit Linkaddress!_init Linkaddress!_register_tm_clones Libc-2.27.so!_setjmp Libc2.27.so!__sigsetjmp Libc2.27.so!__sigjmpsave
Main 函数执行之后	Linkaddress!puts@plt Linkaddress!useless@plt Linkaddress!showpointer@plt malloc Linkaddress!.plt Libc-2.27.so!exi

第 3 章 各阶段的原理与方法

每阶段 40 分，phasex.o 20 分，分析 20 分，总分不超过 80 分

3.1 阶段 1 的分析

程序运行结果截图：

```
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ hexedi
tor
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ gcc -m
32 -o linkbomb1 main.o phase1.o
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ ./link
bomb1
L170300901
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$
```

分析与设计的过程：

首先我们直接进行链接操作，查看结果。

```
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ ././li
nkbomb
YcpwLIvWnRJH8ngvlHwo3Nd8Bvkn6IKvygMueoKQ5Ri4GK YdJ1TTWd8Oo1MyNiNwrjkHLcfBZVL6AX1
fGkX49s5GPHtBUWxooowP31Pp5v4TzPM        0b6J5Ktnb81PbITnrSMIKcJUPiZpJObMV05Ocw8t
GKrUA    ycvPVkODblgOd35WQ44ZzqId
```

然后在 data 字段查找这段乱码

```
00000070  00 00 00 00  00 00 00 00   00 00 00 00  00 00 00 00   ................
00000080  00 00 00 00  00 00 00 00   00 00 00 00  00 44 46 4C   .............DFL
00000090  6A 00 00 4C  31 37 30 33   30 30 39 30  31 00 00 00   j..L170300901...
000000A0  4C 31 37 30  33 30 30 39   30 31 00 00  06 6B 6E 36   L170300901...kn6
000000B0  49 4B 76 79  67 4D 75 65   6F 4B 51 35  52 69 34 47   IKvygMueoKQ5Ri4G
000000C0  4B 20 59 64  4A 31 54 54   57 64 38 4F  6F 31 4D 79   K YdJ1TTWd8Oo1My
000000D0  4E 69 4E 77  72 6A 6B 48   4C 63 66 42  5A 56 4C 36   NiNwrjkHLcfBZVL6
000000E0  41 58 31 66  47 6B 58 34   39 73 35 47  50 48 74 42   AX1fGkX49s5GPHtB
000000F0  55 57 78 6F  6F 6F 77 50   33 31 50 70  35 76 34 54   UWxooowP31Pp5v4T
00000100  7A 50 4D 09  30 62 36 4A   35 4B 74 6E  62 38 31 50   zPM.0b6J5Ktnb81P
00000110  62 49 54 6E  72 53 4D 49   4B 63 4A 55  50 69 5A 70   bITnrSMIKcJUPiZp
00000120  4A 4F 62 4D  56 30 35 4F   63 77 38 74  47 4B 72 55   JObMV05Ocw8tGKrU
00000130  41 09 79 63  76 50 56 6B   4F 44 62 6C  67 4F 64 33   A.ycvPVkODblgOd3
00000140  35 57 51 34  34 5A 7A 71   49 64 00 00  00 00 00 00   5WQ44ZzqId......
```

显然这段就是我们需要修改的部分，我们只需要将此修改为我们的学号即可。

修改如下（我的学号为 L170300901）。

```
00000070  00 00 00 00  00 00 00 00   00 00 00 00  00 00 00 00   ................
00000080  00 00 00 00  00 00 00 00   00 00 00 00  00 44 46 4C   .............DFL
00000090  6A 00 00 4C  31 37 30 33   30 30 39 30  31 00 00 00   j..L170300901...
000000A0  4C 31 37 30  33 30 30 39   30 31 00 00  06 6B 6E 36   L170300901...kn6
000000B0  49 4B 76 79  67 4D 75 65   6F 4B 51 35  52 69 34 47   IKvygMueoKQ5Ri4G
000000C0  4B 20 59 64  4A 31 54 54   57 64 38 4F  6F 31 4D 79   K YdJ1TTWd8Oo1My
000000D0  4E 69 4E 77  72 6A 6B 48   4C 63 66 42  5A 56 4C 36   NiNwrjkHLcfBZVL6
000000E0  41 58 31 66  47 6B 58 34   39 73 35 47  50 48 74 42   AX1fGkX49s5GPHtB
000000F0  55 57 78 6F  6F 6F 77 50   33 31 50 70  35 76 34 54   UWxooowP31Pp5v4T
00000100  7A 50 4D 09  30 62 36 4A   35 4B 74 6E  62 38 31 50   zPM.0b6J5Ktnb81P
00000110  62 49 54 6E  72 53 4D 49   4B 63 4A 55  50 69 5A 70   bITnrSMIKcJUPiZp
00000120  4A 4F 62 4D  56 30 35 4F   63 77 38 74  47 4B 72 55   JObMV05Ocw8tGKrU
00000130  41 09 79 63  76 50 56 6B   4F 44 62 6C  67 4F 64 33   A.ycvPVkODblgOd3
00000140  35 57 51 34  34 5A 7A 71   49 64 00 00  00 00 00 00   5WQ44ZzqId......
```

所以这是结果值钱:

```
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ hexedi
tor
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ gcc -m
32 -o linkbomb1 main.o phase1.o
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ ./link
bomb1
L170300901
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$
```

3.2 阶段 2 的分析

程序运行结果截图：

```
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ gcc -m
32 -o linkbomb2 main.o phase2.o
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ ./link
bomb2
L170300901
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$
```

分析与设计的过程：

首先进行反汇编

```
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ gcc -m
32 -o linkbomb2 main.o
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ gcc -m
32 -o linkbomb2 main.o phase2.o
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ ./link
bomb2
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ objdum
p -s -d phase2.o
```

```
00000000 <IqaJXUtr>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   53                      push   %ebx
   4:   83 ec 04                sub    $0x4,%esp
   7:   e8 fc ff ff ff          call   8 <IqaJXUtr+0x8>
   c:   81 c3 02 00 00 00       add    $0x2,%ebx
  12:   83 ec 08                sub    $0x8,%esp
  15:   8d 83 00 00 00 00       lea    0x0(%ebx),%eax
  1b:   50                      push   %eax
  1c:   ff 75 08                pushl  0x8(%ebp)
  1f:   e8 fc ff ff ff          call   20 <IqaJXUtr+0x20>
  24:   83 c4 10                add    $0x10,%esp
  27:   85 c0                   test   %eax,%eax
  29:   75 10                   jne    3b <IqaJXUtr+0x3b>
  2b:   83 ec 0c                sub    $0xc,%esp
  2e:   ff 75 08                pushl  0x8(%ebp)
  31:   e8 fc ff ff ff          call   32 <IqaJXUtr+0x32>
  36:   83 c4 10                add    $0x10,%esp
  39:   eb 01                   jmp    3c <IqaJXUtr+0x3c>
  3b:   90                      nop
  3c:   8b 5d fc                mov    -0x4(%ebp),%ebx
  3f:   c9                      leave
  40:   c3                      ret
```

```
00000041 <do_phase>:
  41:   55                      push   %ebp
  42:   89 e5                   mov    %esp,%ebp
  44:   e8 fc ff ff ff          call   45 <do_phase+0x4>
  49:   05 01 00 00 00          add    $0x1,%eax
  4e:   90                      nop
  4f:   90                      nop
  50:   90                      nop
  51:   90                      nop
  52:   90                      nop
  53:   90                      nop
  54:   90                      nop
  55:   90                      nop
  56:   90                      nop
  57:   90                      nop
  58:   90                      nop
  59:   90                      nop
  5a:   90                      nop
  5b:   90                      nop
  5c:   90                      nop
  5d:   90                      nop
  5e:   90                      nop
  5f:   90                      nop
  60:   90                      nop
  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:   5d                      pop    %ebp
  70:   c3                      ret
```

我们要做的就是修改这些 nop 指令为我们自己的指令。上面那个乱码函数给了我们很好的示范，因此我们只需修改为如下即可

之后修改重定位规则，使得链接器重定位的是我们的代码，而非上面的乱码函数的代码。定位.rel.text 字段以及其内容

找到目标后进行修改，因为我们只是用到了 puts 函数以及 .rodata 的位置，因此我们直接修改这两处的位置即可

test.c ×

```
lea -0x18b0(%eax),%eax
push %eax
call 0xffffffa6
pop %eax
```

getcode.s

```
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ gcc -m
32 -c getcode.s
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ objdum
p -d getcode.o

getcode.o:     file format elf32-i386


Disassembly of section .text:

00000000 <.text>:
   0:   8d 80 50 e7 ff ff       lea    -0x18b0(%eax),%eax
   6:   50                      push   %eax
   7:   e8 a2 ff ff ff          call   0xffffffae
   c:   58                      pop    %eax
```

```
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ gcc -m32 -o linkbomb2
main.o phase2.o
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ ./linkbomb2
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ hexeditor
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ readelf -a phase2.o
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              REL (Relocatable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x0
  Start of program headers:          0 (bytes into file)
  Start of section headers:          1112 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           0 (bytes)
  Number of program headers:         0
  Size of section headers:           40 (bytes)
  Number of section headers:         19
  Section header string table index: 18
```

```
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ objdum
p -s -d phase2.o

phase2.o:     file format elf32-i386

Contents of section .group:
 0000 01000000 0a000000                    ........
Contents of section .group:
 0000 01000000 0b000000                    ........
Contents of section .text:
 0000 5589e553 83ec04e8 fcffffff 81c30200  U..S............
 0010 000083ec 088d8300 00000050 ff7508e8  ...........P.u..
 0020 fcffffff 83c41085 c0751083 ec0cff75  .........u.....u
 0030 08e8fcff ffff83c4 10eb0190 8b5dfcc9  .............]..
 0040 c35589e5 e8fcffff ff050100 008d8050  .U.............P
 0050 e7ffff50 e8a6ffff ff589090 90909090  ...P.....X......
 0060 90909090 90909090 90909090 9090905d  ...............]
 0070 c3                                   .
Contents of section .rodata:
 0000 4c313730 33303039 303100             L170300901.
Contents of section .data.rel.local:
 0000 00000000                             ....
Contents of section .text.__x86.get_pc_thunk.ax:
 0000 8b0424c3                             ..$.
Contents of section .text.__x86.get_pc_thunk.bx:
 0000 8b1c24c3                             ..$.
Contents of section .comment:
 0000 00474343 3a202855 62756e74 7520372e  .GCC: (Ubuntu 7.
```

所以这是结果值钱：

```
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ gcc -m
32 -o linkbomb2 main.o phase2.o
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ ./link
bomb2
L170300901
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$
```

3.3 阶段 3 的分析

程序运行结果截图：

```
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ gcc -m
32 -c test1.c
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ gcc -m
32 -o linkbomb3 main.o phase3.o test1.o
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ ./link
bomb3
L170300901
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$
```

分析与设计的过程：

说完规则我们来看这个实验。利用 nm 指令，我们可以看到文件中的读好定义

```
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ nm phase3.o
00000100 C ZhcjSNjKZg
         U _GLOBAL_OFFSET_TABLE_
         U __stack_chk_fail_local
00000000 T __x86.get_pc_thunk.bx
00000000 T do_phase
00000000 D phase
         U putchar
```

在结合 readelf 的输出结果不难发现该文件中有一命名为 lOXXBDHjiI 的长度为 256 的字符串没有定义为强符号。既然如此，我们就先写如下 C 文件与之链接

```
char ZhcjSNjKZg[256]="
66666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666661
```

```
gdb-peda$ gdb linkbomb3
Undefined command: "gdb".  Try "help".
gdb-peda$ q
l170300901@l170300901-VirtualBox:~/share/lab5/linklab/linklab-L170300901$ gdb li
nkbomb3
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from linkbomb3...(no debugging symbols found)...done.
gdb-peda$ r
Starting program: /home/l170300901/share/lab5/linklab/linklab-L170300901/linkbom
b3
L170300901
[Inferior 1 (process 2906) exited normally]
Warning: not running or target is remote
gdb-peda$ disas do_phase
Dump of assembler code for function do_phase:
   0x56555605 <+0>:     push   ebp
   0x56555606 <+1>:     mov    ebp,esp
   0x56555608 <+3>:     push   ebx
   0x56555609 <+4>:     sub    esp,0x24
   0x5655560c <+7>:     call   0x565554b0 <__x86.get_pc_thunk.bx>
   0x56555611 <+12>:    add    ebx,0x19bf
   0x56555617 <+18>:    mov    eax,gs:0x14
   0x5655561d <+24>:    mov    DWORD PTR [ebp-0xc],eax
   0x56555620 <+27>:    xor    eax,eax
   0x56555622 <+29>:    mov    DWORD PTR [ebp-0x17],0x77646162
```

```
   0x56555622 <+29>:    mov    DWORD PTR [ebp-0x17],0x77646162
   0x56555629 <+36>:    mov    DWORD PTR [ebp-0x13],0x70637a78
   0x56555630 <+43>:    mov    WORD PTR [ebp-0xf],0x756e
   0x56555636 <+49>:    mov    BYTE PTR [ebp-0xd],0x0
   0x5655563a <+53>:    mov    DWORD PTR [ebp-0x1c],0x0
   0x56555641 <+60>:    jmp    0x5655566e <do_phase+105>
   0x56555643 <+62>:    lea    edx,[ebp-0x17]
   0x56555646 <+65>:    mov    eax,DWORD PTR [ebp-0x1c]
   0x56555649 <+68>:    add    eax,edx
   0x5655564b <+70>:    movzx  eax,BYTE PTR [eax]
   0x5655564e <+73>:    movzx  eax,al
   0x56555651 <+76>:    lea    edx,[ebx+0x50]
   0x56555657 <+82>:    movzx  eax,BYTE PTR [edx+eax*1]
   0x5655565b <+86>:    movsx  eax,al
   0x5655565e <+89>:    sub    esp,0xc
   0x56555661 <+92>:    push   eax
   0x56555662 <+93>:    call   0x56555450 <putchar@plt>
   0x56555667 <+98>:    add    esp,0x10
   0x5655566a <+101>:   add    DWORD PTR [ebp-0x1c],0x1
   0x5655566e <+105>:   mov    eax,DWORD PTR [ebp-0x1c]
   0x56555671 <+108>:   cmp    eax,0x9
   0x56555674 <+111>:   jbe    0x56555643 <do_phase+62>
   0x56555676 <+113>:   sub    esp,0xc
   0x56555679 <+116>:   push   0xa
   0x5655567b <+118>:   call   0x56555450 <putchar@plt>
   0x56555680 <+123>:   add    esp,0x10
   0x56555683 <+126>:   nop
   0x56555684 <+127>:   mov    eax,DWORD PTR [ebp-0xc]
   0x56555687 <+130>:   xor    eax,DWORD PTR gs:0x14
   0x5655568e <+137>:   je     0x56555695 <do_phase+144>
   0x56555690 <+139>:   call   0x56555710 <__stack_chk_fail_local>
   0x56555695 <+144>:   mov    ebx,DWORD PTR [ebp-0x4]
   0x56555698 <+147>:   leave
   0x56555699 <+148>:   ret
End of assembler dump.
gdb-peda$ b*0x5655565b
Breakpoint 1 at 0x5655565b
gdb-peda$ r
Starting program: /home/l170300901/share/lab5/linklab/linklab-L170300901/linkbomb3
```

```
[----------------------------------registers-----------------------------------]
EAX: 0x4c ('L')
EBX: 0x56556fd0 --> 0x1ed8
ECX: 0xffffcf40 --> 0x1
EDX: 0x56557020 ('6' <repeats 97 times>, "1L07666666666606966661603 60", '6' <repeats 77
 times>...)
ESI: 0xf7fb1000 --> 0x1d4d6c
EDI: 0x0
EBP: 0xffffcf18 --> 0xffffcf28 --> 0x0
ESP: 0xffffcef0 --> 0xf7fb1000 --> 0x1d4d6c
EIP: 0x5655565b (<do_phase+86>: movsx  eax,al)
EFLAGS: 0x282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x5655564e <do_phase+73>:    movzx  eax,al
   0x56555651 <do_phase+76>:    lea    edx,[ebx+0x50]
   0x56555657 <do_phase+82>:    movzx  eax,BYTE PTR [edx+eax*1]
=> 0x5655565b <do_phase+86>:    movsx  eax,al
   0x5655565e <do_phase+89>:    sub    esp,0xc
   0x56555661 <do_phase+92>:    push   eax
   0x56555662 <do_phase+93>:    call   0x56555450 <putchar@plt>
   0x56555667 <do_phase+98>:    add    esp,0x10
[-------------------------------------stack-------------------------------------]
0000| 0xffffcef0 --> 0xf7fb1000 --> 0x1d4d6c
0004| 0xffffcef4 --> 0xf7fb1000 --> 0x1d4d6c
0008| 0xffffcef8 --> 0x0
0012| 0xffffcefc --> 0x0
0016| 0xffffcf00 --> 0x646162fc
0020| 0xffffcf04 ("wxzcpnu")
0024| 0xffffcf08 --> 0x756e70 ('pnu')
0028| 0xffffcf0c --> 0x2018100
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0x5655565b in do_phase ()
gdb-peda$ i r
eax            0x4c     0x4c
ecx            0xffffcf40       0xffffcf40
edx            0x56557020       0x56557020
ebx            0x56556fd0       0x56556fd0
esp            0xffffcef0       0xffffcef0
ebp            0xffffcf18       0xffffcf18
esi            0xf7fb1000       0xf7fb1000
edi            0x0      0x0
eip            0x5655565b       0x5655565b <do_phase+86>
eflags         0x282    [ SF IF ]
```

```
eflags         0x282    [ SF IF ]
cs             0x23     0x23
ss             0x2b     0x2b
ds             0x2b     0x2b
es             0x2b     0x2b
fs             0x0      0x0
gs             0x63     0x63
gdb-peda$ disas
Dump of assembler code for function do_phase:
   0x56555605 <+0>:     push   ebp
   0x56555606 <+1>:     mov    ebp,esp
   0x56555608 <+3>:     push   ebx
   0x56555609 <+4>:     sub    esp,0x24
   0x5655560c <+7>:     call   0x565554b0 <__x86.get_pc_thunk.bx>
   0x56555611 <+12>:    add    ebx,0x19bf
   0x56555617 <+18>:    mov    eax,gs:0x14
   0x5655561d <+24>:    mov    DWORD PTR [ebp-0xc],eax
   0x56555620 <+27>:    xor    eax,eax
   0x56555622 <+29>:    mov    DWORD PTR [ebp-0x17],0x77646162
   0x56555629 <+36>:    mov    DWORD PTR [ebp-0x13],0x70637a78
   0x56555630 <+43>:    mov    WORD PTR [ebp-0xf],0x756e
   0x56555636 <+49>:    mov    BYTE PTR [ebp-0xd],0x0
   0x5655563a <+53>:    mov    DWORD PTR [ebp-0x1c],0x0
   0x56555641 <+60>:    jmp    0x5655566e <do_phase+105>
   0x56555643 <+62>:    lea    edx,[ebp-0x17]
   0x56555646 <+65>:    mov    eax,DWORD PTR [ebp-0x1c]
   0x56555649 <+68>:    add    eax,edx
   0x5655564b <+70>:    movzx  eax,BYTE PTR [eax]
   0x5655564e <+73>:    movzx  eax,al
   0x56555651 <+76>:    lea    edx,[ebx+0x50]
   0x56555657 <+82>:    movzx  eax,BYTE PTR [edx+eax*1]
=> 0x5655565b <+86>:    movsx  eax,al
   0x5655565e <+89>:    sub    esp,0xc
   0x56555661 <+92>:    push   eax
   0x56555662 <+93>:    call   0x56555450 <putchar@plt>
   0x56555667 <+98>:    add    esp,0x10
   0x5655566a <+101>:   add    DWORD PTR [ebp-0x1c],0x1
   0x5655566e <+105>:   mov    eax,DWORD PTR [ebp-0x1c]
   0x56555671 <+108>:   cmp    eax,0x9
   0x56555674 <+111>:   jbe    0x56555643 <do_phase+62>
   0x56555676 <+113>:   sub    esp,0xc
   0x56555679 <+116>:   push   0xa
   0x5655567b <+118>:   call   0x56555450 <putchar@plt>
   0x56555680 <+123>:   add    esp,0x10
   0x56555683 <+126>:   nop
```

```
End of assembler dump.
gdb-peda$ x/s $ebp-0x17
0xffffcf01:     "badwxzcpnu"
gdb-peda$
```

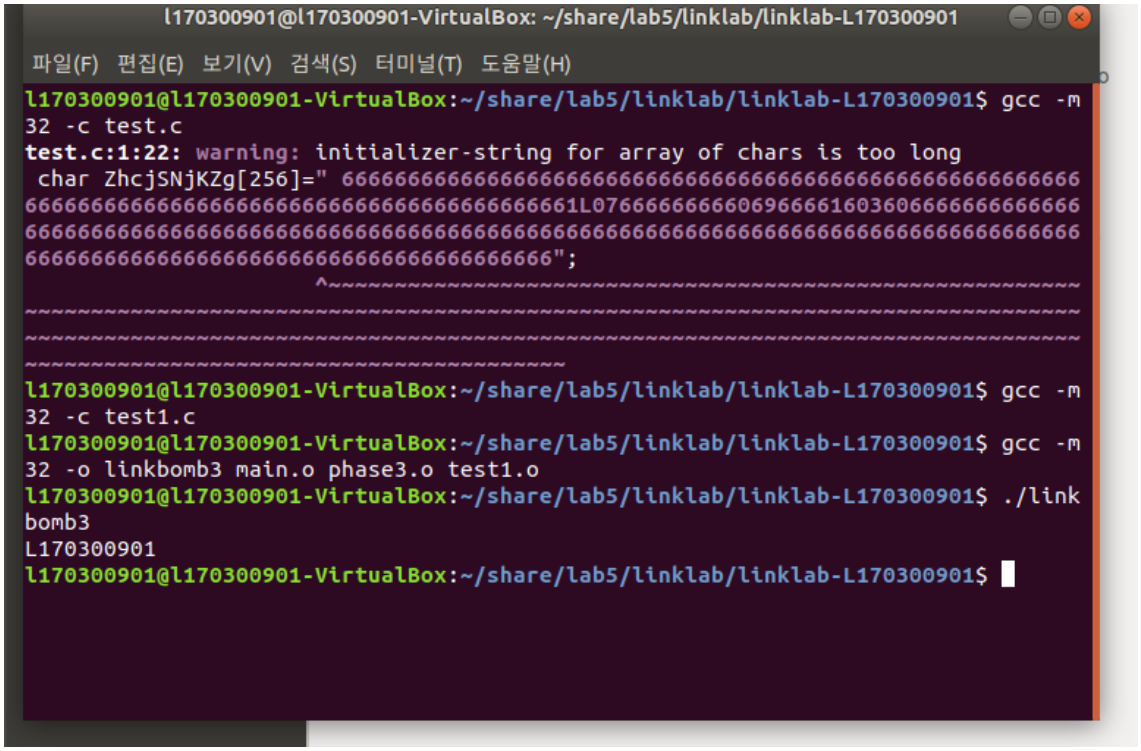
我的 cookie 价格是 “badwxzconu”

我的 cookie 价格出自阿斯基的密码

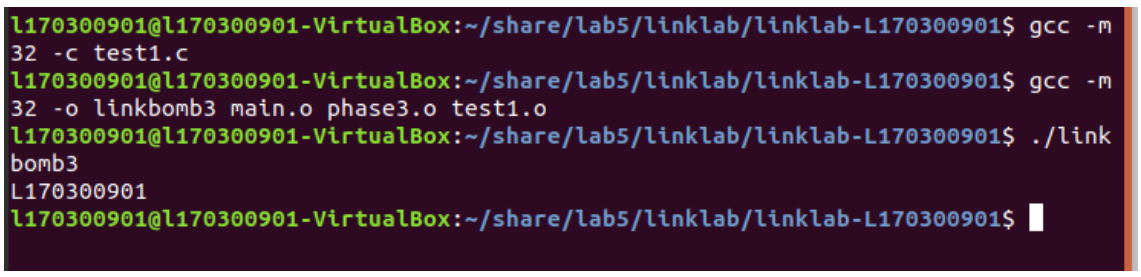
之后,将 Asky Code 的价格改为 10 进位

这就好办多了，我们只需要保证对应的字符为我们的学号就可以啦

因此构造如下 C 文件



所以这是结果值钱



3.4 阶段 4 的分析

程序运行结果截图：

分析与设计的过程：

3.5 阶段 5 的分析

程序运行结果截图：

分析与设计的过程：

第 4 章 总结

4.1 请总结本次实验的收获

我是韩国留学生. 因此用中文学习这个课程是非常困难的. 但是很多中国朋友告诉我和帮助我了解了课堂内容 所以我提高了很多汉语水平,也了解了很多 linux. 对我来说,这似乎是一次很好的经验,真的很幸福

4.2 请给出对本次实验内容的建议

我是留学生 。请多多关照。 谢谢老师
注：本章为酌情加分项。

参考文献

为完成本次实验你翻阅的书籍与网站等

[1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.

[2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A 集[C]. 北京：中国科学出版社，1999.

[3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. http://www.ie.nthu.edu.tw/info/ie.newie.htm（Big5）.

[4] 谌颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.

[5] KANAMORI H. Shaking Without Quaking[J]. Science，1998，279（5359）：2063-2064.

[6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science，1998，281：331-332[1998-09-23]. http://www.sciencemag.org/cgi/collection/anatmorp.