

# 哈尔滨工业大学

# 实验报告

## 实验（五）

题 目 LinkLab

链接

专 业 计算机类

学 号 1170301027

班 级 1703010

学 生 冯帅

指 导 教 师 史先俊

实 验 地 点 G712

实 验 日 期 20181118

计算机科学与技术学院

# 目 录

第 1 章 实验基本信息.....	- 3 -
1.1 实验目的.....	- 3 -
1.2 实验环境与工具.....	- 3 -
1.2.1 硬件环境.....	错误! 未定义书签。
1.2.2 软件环境.....	错误! 未定义书签。
1.2.3 开发工具.....	错误! 未定义书签。
1.3 实验预习.....	- 3 -
第 2 章 实验预习.....	- 4 -
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） .....	- 4 -
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） .....	- 5 -
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 -
（5 分） .....	- 5 -
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） .....	- 6 -
第 3 章 各阶段的原理与方法.....	- 11 -
3.1 阶段 1 的分析.....	- 11 -
3.2 阶段 2 的分析.....	- 12 -
3.3 阶段 3 的分析.....	- 14 -
3.4 阶段 4 的分析.....	- 16 -
3.5 阶段 5 的分析.....	- 17 -
第 4 章 总结.....	- 20 -
4.1 请总结本次实验的收获.....	- 20 -
4.2 请给出对本次实验内容的建议.....	- 20 -
参考文献.....	- 21 -

## 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

Intel Core i7-7700HQ 2.81GHz, 8GB RAM, 128GB SSD

#### 1.2.2 软件环境

Windows10 64 位以上; 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; CodeBlocks 64 位; vi/vim/gedit+gcc

### 1.3 实验预习

填写

## 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

字大小、字节顺序、文件类型(.o, exec, .so), 机器类型, 等等

段头表/程序头表

页面大小, 虚拟地址内存段(节), 段大小

.text 节 (代码)

代码

.rodata 节 (只读数据)

只读数据 : 跳转表, ...

.data 节 (数据/可读写)

已初始化全局变量

.bss 节 (未初始化全局变量)

未初始化的全局变量

“Block Started by Symbol” 符号开始的块

“Better Save Space” 更加节省空间

有节头, 但不占用空间

.symtab 节 (符号表)

符号表

函数和静态变量名

节名称和位置

.rel.text 节 (可重定位代码)

.text 节的可重定位信息

在可执行文件中需要修改的指令地址

需修改的指令.

.rel.data 节 (可重定位数据)

.data 节的可重定位信息

在合并后的可执行文件中需要修改的指针数据的地址

.debug 节 (调试)

为符号调试的信息 (gcc -g)

节头表 Section header table

每个节的偏移量和大小

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)

0x400000

只读代码段  
(.init, .text, .rodata)

读写数据段  
(.data, .bss)

运行时 堆 (heap)  
(由 malloc 动态生成<128K)

共享库的内存映射区域  
共享内存 (mmap)

用户栈 (User stack)  
运行时创建

内核虚存区

$2^{48}-1$ ;

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)

读写数据段  
(.data, .bss)

big array 0x55a4df0010e0 94166604320992

huge array 0x55a49f0010e0 94165530579168

global 0x55a49f001020 94165530578976

gint0 0x55a49f0010cc 94165530579148

glong 0x55a49f001028 94165530578984

cstr 0x55a49f001040 94165530579008

只读代码段

(.init, .text, .rodata)

pstr 0x55a49edfffa0 94165528477600

gc 0x55a49edfffcc 94165528477644

cc 0x55a49edfffe0 94165

show\_pointer 0x55a49edff875 94165528475765

useless 0x55a49edff86a 94165528475754

main 0x55a49edff8a8 94165528475816528477664

local static int 0 0x55a49f0010d0 94165530579152

local static int 1 0x55a49f0010a4 94165530579108

local str 0x7ffedcb7f4d0 140732601464016

argc 0x7ffedcb7f48c 140732601463948

argv 0x7ffedcb7f9b8 140732601465272

argv[0] 7ffedcb802de

argv[1] 7ffedcb802e6

argv[2] 7ffedcb802e9

argv[3] 7ffedcb802f4

argv[0] 0x7ffedcb802de 140732601467614

./a.out

argv[1] 0x7ffedcb802e6 140732601467622

-u

argv[2] 0x7ffedcb802e9 140732601467625

1170301027

argv[3] 0x7ffedcb802f4 140732601467636

冯帅

共享库的内存映射区域

共享内存 (mmap)

用户栈 (User stack)

p1 0x7fe8b6d61010 140637476622352  
p2 0x55a4e1db267094166652233328//heap  
p3 0x7fe8c733f010 140637751210000  
p4 0x7fe876d60010 140636402876432  
p5 (nil)0  
local int 0 0x7ffedcb7f494 140732601463956  
local int 1 0x7ffedcb7f498 140732601463960

内核虚存区

exit 0x7fe8c6da5120 140637745336608  
printf 0x7fe8c6dc6e80 140637745475200  
malloc 0x7fe8c6df9070 140637745680496  
free 0x7fe8c6df9950 140637745682768  
strcpy 0x7fe8c6e18540 140637745808704

env 0x7ffedcb7f9e0 140732601465312  
env[0] \*env 0x7ffedcb802fb 140732601467643  
CLUTTER\_IM\_MODULE=xim  
env[1] \*env 0x7ffedcb80311 140732601467665  
env[2] \*env 0x7ffedcb808fd 140732601469181  
LC\_MEASUREMENT=zh\_CN.UTF-8  
env[3] \*env 0x7ffedcb80918 140732601469208  
LESSCLOSE=/usr/bin/lesspipe %s %s  
env[4] \*env 0x7ffedcb8093a 140732601469242  
LC\_PAPER=zh\_CN.UTF-8  
env[5] \*env 0x7ffedcb8094f 140732601469263  
LC\_MONETARY=zh\_CN.UTF-8  
env[6] \*env 0x7ffedcb80967 140732601469287  
XDG\_MENU\_PREFIX=gnome-  
env[7] \*env 0x7ffedcb8097e 140732601469310  
LANG=zh\_CN.UTF-8  
env[8] \*env 0x7ffedcb8098f 140732601469327  
DISPLAY=:0  
env[9] \*env 0x7ffedcb8099a 140732601469338  
GNOME\_SHELL\_SESSION\_MODE=ubuntu  
env[10] \*env 0x7ffedcb809ba 140732601469370  
COLORTERM=truecolor

```
env[11] *env 0x7ffedcb809ce 140732601469390
USERNAME=fs1170301027
env[12] *env 0x7ffedcb809e4 140732601469412
XDG_VTNR=2
env[13] *env 0x7ffedcb809ef 140732601469423
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
env[14] *env 0x7ffedcb80a18 140732601469464
LC_NAME=zh_CN.UTF-8
env[15] *env 0x7ffedcb80a2c 140732601469484
XDG_SESSION_ID=2
env[16] *env 0x7ffedcb80a3d 140732601469501
USER=fs1170301027
env[17] *env 0x7ffedcb80a4f 140732601469519
DESKTOP_SESSION=ubuntu
env[18] *env 0x7ffedcb80a66 140732601469542
QT4_IM_MODULE=xim
env[19] *env 0x7ffedcb80a78 140732601469560
TEXTDOMAINDIR=/usr/share/locale/
env[20] *env 0x7ffedcb80a99 140732601469593
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/c8f559b1_85da_48f9_b
fa2_b3e2bdd33b71
env[21] *env 0x7ffedcb80aef 140732601469679
PWD=/home/fs1170301027/hitics/linklab
env[22] *env 0x7ffedcb80b15 140732601469717
HOME=/home/fs1170301027
env[23] *env 0x7ffedcb80b2d 140732601469741
TEXTDOMAIN=im-config
env[24] *env 0x7ffedcb80b42 140732601469762
SSH_AGENT_PID=1990
env[25] *env 0x7ffedcb80b55 140732601469781
QT_ACCESSIBILITY=1
env[26] *env 0x7ffedcb80b68 140732601469800
XDG_SESSION_TYPE=x11
env[27] *env 0x7ffedcb80b7d 140732601469821
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd/desktop
env[28] *env 0x7ffedcb80bd0 140732601469904
XDG_SESSION_DESKTOP=ubuntu
env[29] *env 0x7ffedcb80beb 140732601469931
LC_ADDRESS=zh_CN.UTF-8
env[30] *env 0x7ffedcb80c02 140732601469954
GJS_DEBUG_OUTPUT=stderr
env[31] *env 0x7ffedcb80c1a 140732601469978
LC_NUMERIC=zh_CN.UTF-8
env[32] *env 0x7ffedcb80c31 140732601470001
GTK_MODULES=gail:atk-bridge
env[33] *env 0x7ffedcb80c4d 140732601470029
PAPERSIZE=a4
```



```
env[34] *env 0x7ffedcb80c5a 140732601470042
WINDOWPATH=2
env[35] *env 0x7ffedcb80c67 140732601470055
TERM=xterm-256color
env[36] *env 0x7ffedcb80c7b 140732601470075
SHELL=/bin/bash
env[37] *env 0x7ffedcb80c8b 140732601470091
VTE_VERSION=5202
env[38] *env 0x7ffedcb80c9c 140732601470108
QT_IM_MODULE=ibus
env[39] *env 0x7ffedcb80cae 140732601470126
XMODIFIERS=@im=ibus
env[40] *env 0x7ffedcb80cc2 140732601470146
IM_CONFIG_PHASE=2
env[41] *env 0x7ffedcb80cd4 140732601470164
XDG_CURRENT_DESKTOP=ubuntu:GNOME
env[42] *env 0x7ffedcb80cf5 140732601470197
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
env[43] *env 0x7ffedcb80d29 140732601470249
GNOME_TERMINAL_SERVICE=:1.82
env[44] *env 0x7ffedcb80d46 140732601470278
XDG_SEAT=seat0
env[45] *env 0x7ffedcb80d55 140732601470293
SHLVL=1
env[46] *env 0x7ffedcb80d5d 140732601470301
LANGUAGE=zh_CN:en_US:en
env[47] *env 0x7ffedcb80d75 140732601470325
LC_TELEPHONE=zh_CN.UTF-8
env[48] *env 0x7ffedcb80d8e 140732601470350
GDMSESSION=ubuntu
env[49] *env 0x7ffedcb80da0 140732601470368
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
env[50] *env 0x7ffedcb80dcc 140732601470412
LOGNAME=fs1170301027
env[51] *env 0x7ffedcb80de1 140732601470433
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
env[52] *env 0x7ffedcb80e17 140732601470487
XDG_RUNTIME_DIR=/run/user/1000
env[53] *env 0x7ffedcb80e36 140732601470518
XAUTHORITY=/run/user/1000/gdm/Xauthority
env[54] *env 0x7ffedcb80e5f 140732601470559
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
env[55] *env 0x7ffedcb80e8c 140732601470604
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/ga
mes:/snap/bin
env[56] *env 0x7ffedcb80ef4 140732601470708
LC_IDENTIFICATION=zh_CN.UTF-8
```

```
env[57] *env 0x7ffedcb80f12 140732601470738
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
env[58] *env 0x7ffedcb80f33 140732601470771
SESSION_MANAGER=local/ubuntu:@/tmp/.ICE-unix/1912,unix/ubuntu:/tmp/.ICE-unix/1912
env[59] *env 0x7ffedcb80f85 140732601470853
LESSOPEN=| /usr/bin/lesspipe %s
env[60] *env 0x7ffedcb80fa5 140732601470885
GTK_IM_MODULE=ibus
env[61] *env 0x7ffedcb80fb8 140732601470904
LC_TIME=zh_CN.UTF-8
env[62] *env 0x7ffedcb80fcc 140732601470924
OLDPWD=/home/fs1170301027
env[63] *env 0x7ffedcb80fe6 140732601470950
_=./a.out
```

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)

Main 函数执行前:

```
Ld-2.27.so!_dl_start
Ld-2.27.so!_dl_init
Libc-2.27.so!_dl_cxa_atexit
Linkaddress!_init
Linkaddress!_register_tm_clones
Libc-2.27.so!_dl_setjmp
Libc-2.27.so!_dl_sigsetjmp
Libc-2.27.so!_dl_sigjmpsave
```

Main 函数执行后

```
Linkaddress!puts@plt
Linkaddress!useless@plt
Linkaddress!showpointer@plt
Malloc
Linkaddress!.plt
Libc-2.27.so!exit
```

## 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

我做的是第二个包下的，

### 3.1 阶段 1 的分析

程序运行结果截图：

```
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 main.o phase1.o -o p1
fs1170301027@ubuntu:~/hitics/linklab1$ ./p1
1170301027
```

分析与设计的过程：

```
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 main.o phase1.o -o p1
fs1170301027@ubuntu:~/hitics/linklab1$ ./p1
t00IaDtGc          vQy5W7Y0uuxlIjqoMY0JlZupTQ ddpvv6293zXHnrMzocd7egL4DWJqUc0edYaa
1MA tCApeNZd6a5dwoCFTVii9mU
```

```

fs1170301027@ubuntu: ~/hitics/linklab1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
00000000 7F 45 4C 46 01 01 01 00 00 00 00 00 00 00 00 00 .ELF.....
00000010 01 00 03 00 01 00 00 00 00 00 00 00 00 00 00 00 .....
00000020 8C 02 00 00 00 00 00 00 34 00 00 00 00 00 28 00 .....4.....(
00000030 0D 00 0C 00 55 89 E5 83 EC 08 83 EC 0C 68 00 00 ....U.....h..
00000040 00 00 E8 FC FF FF FF 83 C4 10 90 C9 C3 00 00 00 .....
00000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000060 74 4F 30 49 61 44 74 67 43 09 76 51 79 35 57 37 t00IaDtgc.vQy5W7
00000070 59 4F 75 75 78 6C 49 6A 71 6F 4D 59 30 4A 6C 5A Y0uuxlijqoMY0JLZ
00000080 75 70 54 51 20 64 64 70 76 76 36 32 39 33 7A 58 upTQ ddpvv6293zX
00000090 48 6E 72 4D 7A 6F 63 64 37 65 67 4C 34 44 57 4A HnrMzocd7egL4DWJ
000000A0 71 55 63 30 65 64 59 61 61 31 4D 41 20 74 43 41 qUc0edYaa1MA tCA
000000B0 70 65 4E 5A 64 36 61 35 64 77 6F 43 46 54 56 49 peNZd6a5dwoCFTVI
000000C0 69 39 6D 55 00 00 00 00 00 00 00 00 00 47 43 43 i9mU.....GCC
000000D0 3A 20 28 55 62 75 6E 74 75 20 37 2E 33 2E 30 2D : (Ubuntu 7.3.0-
000000E0 31 36 75 62 75 6E 74 75 33 29 20 37 2E 33 2E 30 16ubuntu3) 7.3.0
000000F0 00 00 00 00 14 00 00 00 00 00 00 00 01 7A 52 00 .....zR.
00000100 01 7C 08 01 1B 0C 04 04 88 01 00 00 1C 00 00 00 .|.
00000110 1C 00 00 00 00 00 00 00 19 00 00 00 00 41 0E 08 .....A..
00000120 85 02 42 0D 05 55 C5 0C 04 04 00 00 00 00 00 00 ..B..U.....
00000130 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 .....
00000140 00 00 00 00 00 00 00 00 04 00 F1 FF 00 00 00 00 .....
00000150 00 00 00 00 00 00 00 00 03 00 01 00 00 00 00 00 .....
00000160 00 00 00 00 00 00 00 00 03 00 03 00 00 00 00 00 .....
00000170 00 00 00 00 00 00 00 00 03 00 05 00 0A 00 00 00 .....
00000180 00 00 00 00 65 00 00 00 01 00 03 00 00 00 00 00 ....e.....
00000190 00 00 00 00 00 00 00 00 03 00 07 00 00 00 00 00 .....
000001A0 00 00 00 00 00 00 00 00 03 00 08 00 00 00 00 00 .....
--- phase1.o --0x60/0x494-----

```

要将输入定向到该虚拟机，请将鼠标指针移入其中或按 Ctrl+G。

```
00000060 31 31 37 30 33 30 31 30 32 37 00 51 79 35 57 37 1170301027.Qy5W7
```

对比于更改之前的输出字符串，用 hexedit 索引到相应输出的位置直接修改为学号的 hex 保存就成功了

## 3.2 阶段 2 的分析

程序运行结果截图：

```

fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 -no-pie -fno-PIC main.o phase2.o -o p2
fs1170301027@ubuntu:~/hitics/linklab1$ ./p2
1170301027

```

分析与设计的过程：

```

00000030 <do_phase>:
 30: 55                push    %ebp
 31: 89 e5             mov     %esp,%ebp
 33: e8 c8 ff ff ff   call    0 <nYxhjsJX>
 38: 83 ec 1c          sub     $0x1c,%esp
 3b: e8 c0 ff ff ff   call    0 <nYxhjsJX>
 40: 83 c4 1c          add     $0x1c,%esp
 43: 90                nop
 44: 90                nop
 45: 90                nop
 46: 90                nop

```

从 33 到 40 是更改由 hexedit 看到的，首先通过第一次的 pc 相对寻址调用函数，但是此时肯定是不能输出学号的，分析函数

```

00000000 <nYxhjsJX>:
 0: 55                push    %ebp
 1: 89 e5             mov     %esp,%ebp
 3: 83 ec 08          sub     $0x8,%esp
 6: 83 ec 08          sub     $0x8,%esp
 9: 68 00 00 00 00   push    $0x0
 e: ff 75 08          pushl   0x8(%ebp)
11: e8 fc ff ff ff   call    12 <nYxhjsJX+0x12>
16: 83 c4 10          add     $0x10,%esp
19: 85 c0             test    %eax,%eax
1b: 75 10             jne     2d <nYxhjsJX+0x2d>
1d: 83 ec 0c          sub     $0xc,%esp
20: ff 75 08          pushl   0x8(%ebp)
23: e8 fc ff ff ff   call    24 <nYxhjsJX+0x24>
28: 83 c4 10          add     $0x10,%esp
2b: eb 01             jmp     2e <nYxhjsJX+0x2e>
2d: 90                nop
2e: c9                leave
2f: c3                ret

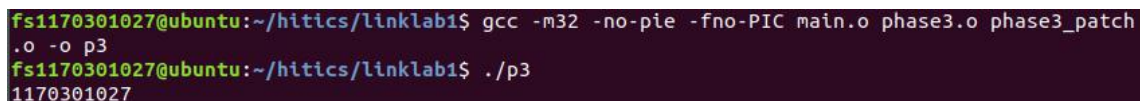
```

返回地址
Push ebp
Esp-8
Esp-8
Push 0
Push ebp+8(参数)

第一遍运行的时候，通过 `leave` 操作恢复栈帧，假设上述栈帧结构每一行代表 `0x4` 的话，此时想寻址到 `push 0` 进行传参就相当于传参 `esp-0x1c` 处的值，那经过该操作之后再次调用函数即能输出正确的学号，最后恢复栈帧即可。

### 3.3 阶段 3 的分析

程序运行结果截图：



```
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 -no-pie -fno-PIC main.o phase3.o phase3_patch.o -o p3
fs1170301027@ubuntu:~/hitics/linklab1$ ./p3
1170301027
```

分析与设计的过程：

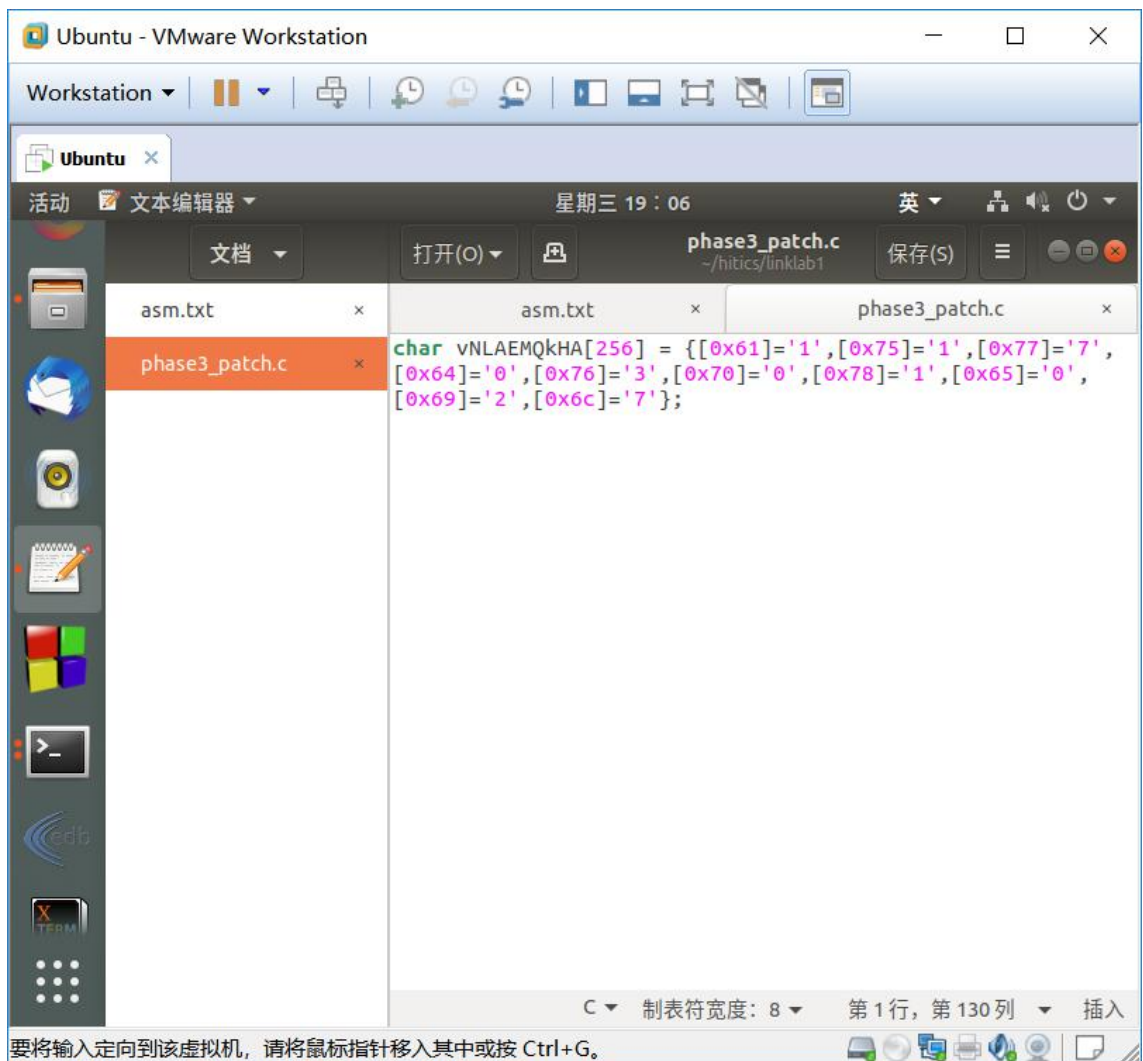


```

无标题 - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
0:      55      push %ebp
1:      89 e5    mov  %esp,%ebp
3:      83 ec 28  sub  $0x28,%esp
6:      65 a1 14 00 00 00    mov  %gs:0x14,%eax
c:      89 45 f4    mov  %eax,-0xc(%ebp)
f:      31 c0      xor  %eax,%eax
11:     c7 45 e9 61 75 77 64    movl $0x64777561,-0x17(%ebp)//23
18:     c7 45 ed 76 70 78 65    movl $0x65787076,-0x13(%ebp)//19
1f:     66 c7 45 f1 69 6c    movw $0x6c69,-0xf(%ebp)//15
25:     c6 45 f3 00    movb $0x0,-0xd(%ebp)//13
29:     c7 45 e4 00 00 00 00    movl $0x0,-0x1c(%ebp)//28
30:     eb 28      jmp  5a <do_phase+0x5a>
32:     8d 55 e9      lea  -0x17(%ebp),%edx/<--
35:     8b 45 e4      mov  -0x1c(%ebp),%eax
38:     01 d0      add  %edx,%eax
3a:     0f b6 00      movzbl (%eax),%eax
3d:     0f b6 c0      movzbl %al,%eax
40:     0f b6 80 00 00 00 00    movzbl 0x0(%eax),%eax
43: R_386_32      vNLAEMQkHA
47:     0f be c0      movsbl %al,%eax
4a:     83 ec 0c      sub  $0xc,%esp
4d:     50      push %eax
4e:     e8 fc ff ff ff    call 4f <do_phase+0x4f>
4f: R_386_PC32    putchar
53:     83 c4 10      add  $0x10,%esp
56:     83 45 e4 01    addl $0x1,-0x1c(%ebp)
5a:     8b 45 e4      mov  -0x1c(%ebp),%eax//计数<--
5d:     83 f8 09      cmp  $0x9,%eax
60:     76 d0      jbe  32 <do_phase+0x32>//eax<9跳转
62:     83 ec 0c      sub  $0xc,%esp
65:     6a 0a      push $0xa
67:     e8 fc ff ff ff    call 68 <do_phase+0x68>
68: R_386_PC32    putchar
6c:     83 c4 10      add  $0x10,%esp
6f:     90      nop

```

分析明白循环之后就好看一点了，其实从 11 到 29 行分别将在数组中的偏移量，也就是数组下标放到了连续的字节中，然后看后续的循环中一次取一个字节作为偏移量输出出来，明白了这一点之后就好写 c 程序了，从 objdump-r 看到的反汇编中，可见 vNLAEMQkHA 即为数组名，又知道所有的偏移量的访问顺序，所以可得 phase3\_patch.c 中直接赋初值为学号即可



继而经过编译，即能得到如下结果，

```
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 -c phase3_patch.c -o phase3_patch.o
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 main.o phase3.o phase3_patch.o -o p3
fs1170301027@ubuntu:~/hitics/linklab1$ ./p3
1170301027
```

### 3.4 阶段 4 的分析

程序运行结果截图：

```
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 -no-pie -fno-PIC main.o phase4.o -o p4
fs1170301027@ubuntu:~/hitics/linklab1$ ./p4
1170301027
```

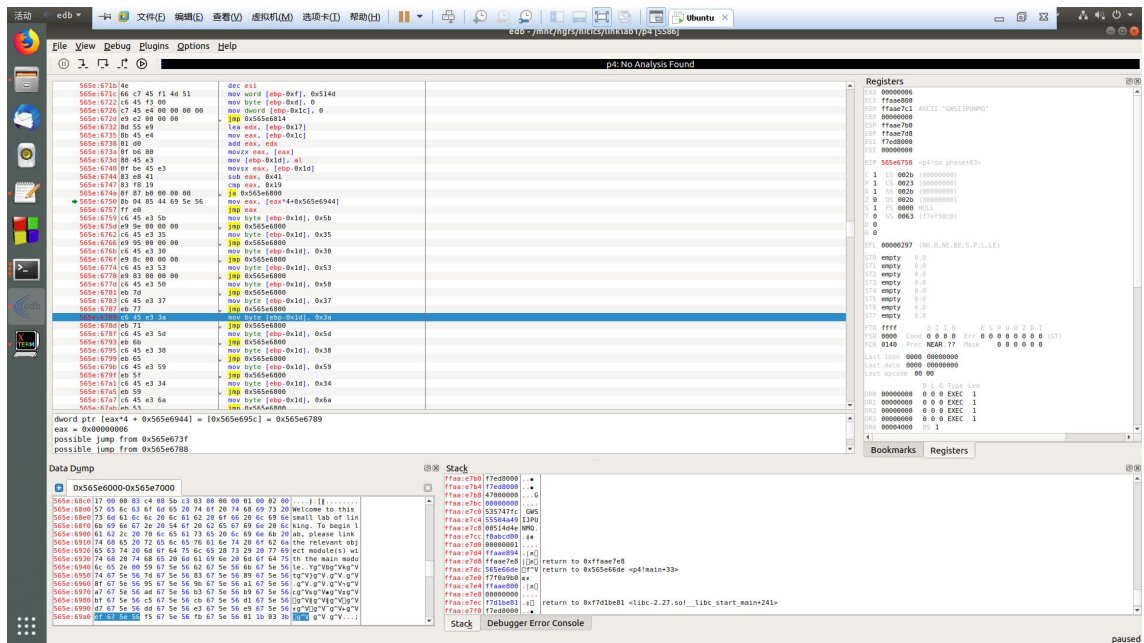
分析与设计的过程：大体思想和上一阶段一样，我只在 edb 中走了一遍，发



现输出的都是 case 之后的 ascii 码值，那好我就直接在 phase4 的 hexedit 中改对应的 ascii 码值就行，

3a 36 7e 38 59 3f 33 7d 32 60----->> 31 31 37 30 33 30 31 30 32 37

附两张过程中的图



### 3.5 阶段 5 的分析

程序运行结果截图：

```
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 -no-pie -fno-PIC main.o phase5.o -o p5
fs1170301027@ubuntu:~/hitics/linklab1$ ./p5
ss[,F,s,5]
```

分析与设计的过程：

```

fs1170301027@ubuntu: ~/hitics/linklab1
文件(E) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
000004A0 55 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 U.....
000004B0 4F 00 00 00 0C 00 00 00 04 00 00 00 11 00 03 00 O.....
000004C0 00 70 68 61 73 65 35 2E 63 00 50 42 51 61 57 44 .phase5.c.PBQaWD
000004D0 00 58 59 49 4E 6E 79 00 42 55 46 00 43 4F 44 45 .XYINny.BUF.CODE
000004E0 00 74 72 61 6E 73 66 6F 72 6D 5F 63 6F 64 65 00 .transform_code.
000004F0 67 65 6E 65 72 61 74 65 5F 63 6F 64 65 00 65 6E generate_code.en
00000500 63 6F 64 65 00 73 74 72 6C 65 6E 00 64 6F 5F 70 code.strlen.do_p
00000510 68 61 73 65 00 70 75 74 73 00 00 00 09 00 00 00 hase.puts.....
00000520 01 09 00 00 18 00 00 00 01 05 00 00 29 00 00 00 .....))...
00000530 01 09 00 00 3D 00 00 00 01 09 00 00 4E 00 00 00 ....=.....N...
00000540 01 09 00 00 60 00 00 00 01 09 00 00 6F 00 00 00 ....`.....O...
00000550 01 09 00 00 8B 00 00 00 01 0C 00 00 9B 00 00 00 .....
00000560 01 0C 00 00 A7 00 00 00 02 0D 00 00 AF 00 00 00 .....
00000570 01 0C 00 00 CF 00 00 00 02 10 00 00 F3 00 00 00 .....
00000580 01 0A 00 00 FA 00 00 00 01 0C 00 00 57 01 00 00 .....W...
00000590 02 0E 00 00 62 01 00 00 01 0B 00 00 67 01 00 00 ....b.....g...
000005A0 02 0F 00 00 72 01 00 00 01 0B 00 00 77 01 00 00 ....r.....w...
000005B0 02 12 00 00 0C 00 00 00 01 11 00 00 C0 00 00 00 .....
000005C0 01 02 00 00 C4 00 00 00 01 02 00 00 C8 00 00 00 .....
000005D0 01 02 00 00 CC 00 00 00 01 02 00 00 D0 00 00 00 .....
000005E0 01 02 00 00 D4 00 00 00 01 02 00 00 D8 00 00 00 .....
000005F0 01 02 00 00 DC 00 00 00 01 02 00 00 20 00 00 00 .....
00000600 02 02 00 00 40 00 00 00 02 02 00 00 60 00 00 00 ....@.....
--- phase5.o --0x5C2/0x8DC-----

```

```

fs1170301027@ubuntu: ~/hitics/linklab1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
重定位节 '.rel.text' at offset 0x51c contains 19 entries:
偏移量 信息 类型 符号值 符号名称
00000009 00000901 R_386_32 00000000 PBQaWD
00000018 00000501 R_386_32 00000000 .rodata
00000029 00000901 R_386_32 00000000 PBQaWD
0000003d 00000901 R_386_32 00000000 PBQaWD
0000004e 00000901 R_386_32 00000000 PBQaWD
00000060 00000901 R_386_32 00000000 PBQaWD
0000006f 00000901 R_386_32 00000000 PBQaWD
0000008b 00000c01 R_386_32 0000000b CODE
0000009b 00000c01 R_386_32 0000000b CODE
000000a7 00000d02 R_386_PC32 00000000 transform_code
000000af 00000c01 R_386_32 0000000b CODE
000000cf 00001002 R_386_PC32 00000000 strlen
000000f3 00000a01 R_386_32 00000040 XYINny
000000fa 00000c01 R_386_32 0000000b CODE
00000157 00000e02 R_386_PC32 00000081 generate_code
00000162 00000b01 R_386_32 00000000 BUF
00000167 00000f02 R_386_PC32 000000c2 encode
00000172 00000b01 R_386_32 00000000 BUF
00000177 00001202 R_386_PC32 00000000 puts

重定位节 '.rel.data' at offset 0x5b4 contains 1 entry:
偏移量 信息 类型 符号值 符号名称

```

通过反汇编语句把重定位节“.rel.text”“.rel.rodata”改了之后就能正确输出

了

无标题 - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
00000009 00000901PBQaWD
0000003d 00000901PBQaWD
0000004e 00000901 PBQaWD
0000008b 000000c01 code
000000f3 00000d01 XYINny
00000157 00000d02 generate_code
00000167 00000d02 encode
00000172 00000b01 buf|
```

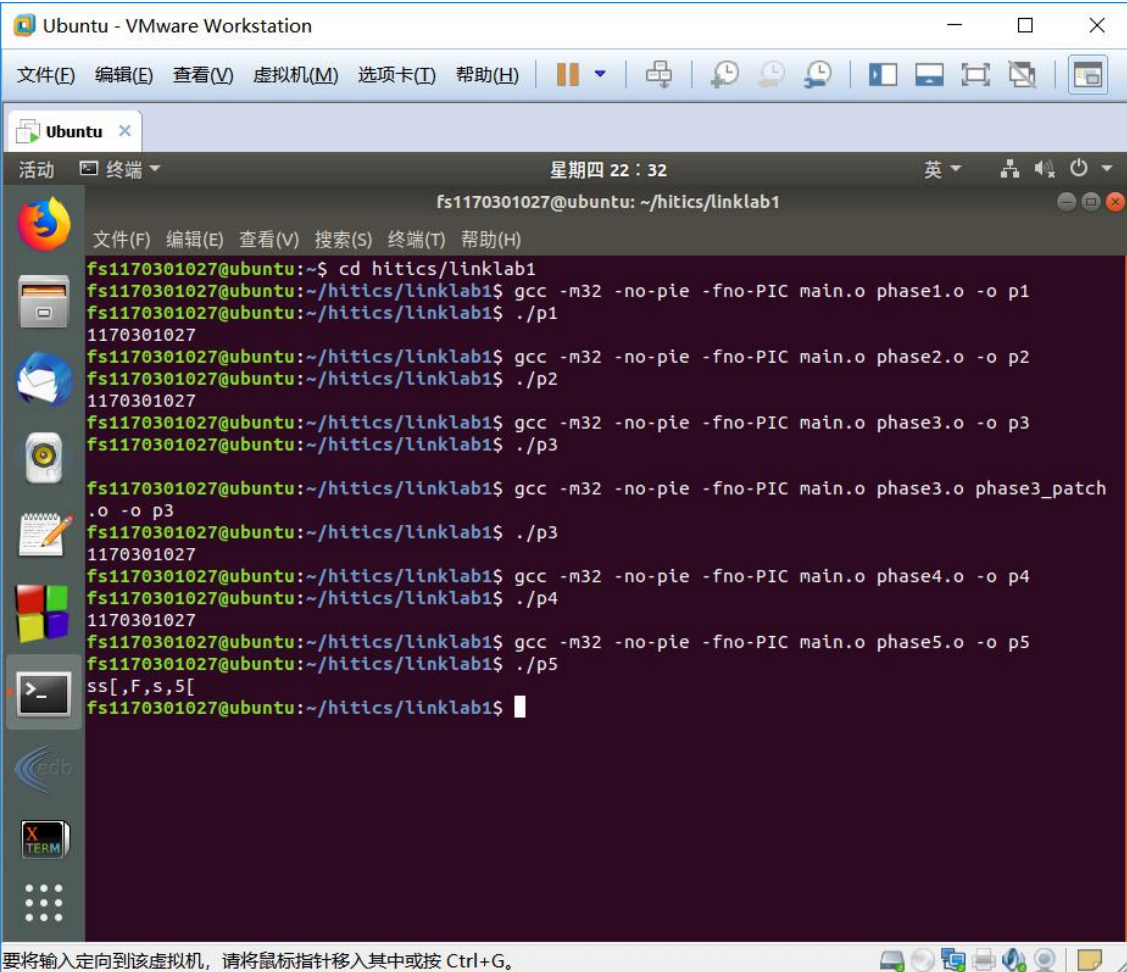




## 第4章 总结

### 4.1 请总结本次实验的收获

鸽了一周再回来煞心看看之后发现其实只是自己没学到位罢了，并不是别人有多强，本次实验让我对链接这部分内容有了一定的了解，可以继续进行下一部分的学习了



```
fs1170301027@ubuntu: ~/hitics/linklab1
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 -no-pie -fno-PIC main.o phase1.o -o p1
fs1170301027@ubuntu:~/hitics/linklab1$ ./p1
1170301027
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 -no-pie -fno-PIC main.o phase2.o -o p2
fs1170301027@ubuntu:~/hitics/linklab1$ ./p2
1170301027
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 -no-pie -fno-PIC main.o phase3.o -o p3
fs1170301027@ubuntu:~/hitics/linklab1$ ./p3
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 -no-pie -fno-PIC main.o phase3.o phase3_patch.o -o p3
fs1170301027@ubuntu:~/hitics/linklab1$ ./p3
1170301027
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 -no-pie -fno-PIC main.o phase4.o -o p4
fs1170301027@ubuntu:~/hitics/linklab1$ ./p4
1170301027
fs1170301027@ubuntu:~/hitics/linklab1$ gcc -m32 -no-pie -fno-PIC main.o phase5.o -o p5
fs1170301027@ubuntu:~/hitics/linklab1$ ./p5
ss[,F,s,5[
fs1170301027@ubuntu:~/hitics/linklab1$
```

### 4.2 请给出对本次实验内容的建议

没什么更好的建议

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>.