

# 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 1190201816

班 级 1903012

学 生 樊红雨

指 导 教 师 史先俊

实 验 地 点 G709

实 验 日 期 2021.5.19

## 计算机科学与技术学院

# 目 录

<b>第 1 章 实验基本信息</b> .....	<b>- 3 -</b>
1.1 实验目的.....	- 3 -
1.2 实验环境与工具.....	- 3 -
1.2.1 硬件环境.....	- 3 -
1.2.2 软件环境.....	- 3 -
1.2.3 开发工具.....	- 3 -
1.3 实验预习.....	- 3 -
<b>第 2 章 实验预习</b> .....	<b>- 4 -</b>
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） .....	- 4 -
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） .....	- 13 -
<b>第 3 章 各阶段的原理与方法</b> .....	<b>- 15 -</b>
3.1 阶段 1 的分析.....	- 15 -
3.2 阶段 2 的分析 .....	- 16 -
3.3 阶段 3 的分析 .....	- 18 -
3.4 阶段 4 的分析 .....	- 21 -
3.5 阶段 5 的分析 .....	- 21 -
<b>第 4 章 总结</b> .....	<b>- 22 -</b>
4.1 请总结本次实验的收获.....	- 22 -
4.2 请给出对本次实验内容的建议.....	- 22 -
<b>参考文献</b> .....	<b>- 23 -</b>

## 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、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”按地址顺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)

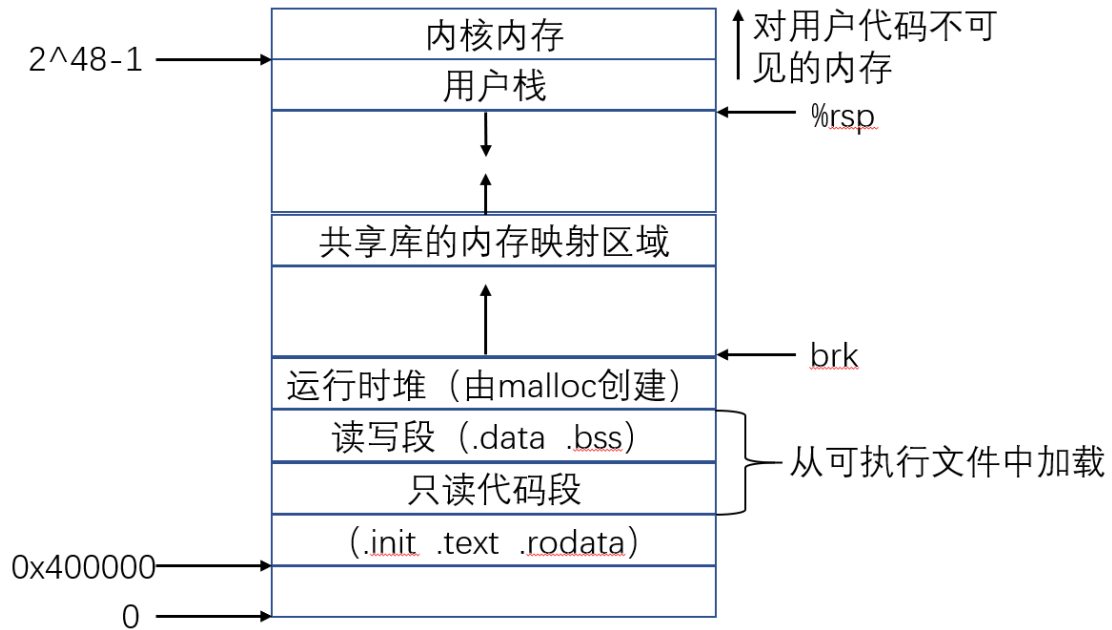
ELF 头

段头部表：将连续的文件映射到运行时的内存段

- . init : 定义了 `_init` 函数，程序初始化代码会调用它
- . text : 已编译程序的机器代码
- . rodata : 只读数据，比如 `printf` 语句中的格式串和开关语句的跳转表
- . data : 已初始化的全局和静态 C 变量
- . bss : 未初始化的全局和静态 C 变量
- . symtab : 一个符号表，它存放在程序中定义和引用的函数和全局变量的信息
- . debug : 一个调试符号表，其条目时程序中定义的全局变量和类型定义，程序中定义和引用的全局变量，以及原始的 C 源文件。
- . line : 原始 C 源程序的行号和 `.text` 节中机器指令之间的映射
- . strtabs : 一个字符串表，其内容包括 `.symtab` 和 `.debug` 节中的符号表，以及节头部中的节名字。

节头部表：描述目标文件的节。

### 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)



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)

所属区	个股好的地址、空间（地址从小到大）
只读代码段 (.init .text .redat a)	exit 0x401100 4198656 printf 0x4010e0 4198624 malloc 0x4010f0 4198640 free 0x4010a0 4198560 strcpy 0x4010b0 4198576
读写段 (.dara .bss)	show_pointer 0x401205 4198917 useless 0x4011f6 4198902

	main 0x40123a 4198970 global 0x404080 4210816 big array 0x404140 4211008 huge array 0x1404140 20988224
运行时堆 (由 malloc 创建)	p1 0x7face78c9010 140380595851280 p2 0x4331b6b0 1127331504 p3 0x7face78a8010 140380595716112 p4 0x7faca78a7010 140379521970192 p5 0x7fac278a6010 140377374482448
用户栈 (运行时创建)	env 0x7ffe743b04f0 140730848445680 env[0] *env 0x7ffe743b12e6 140730848449254 SHELL=/bin/bash env[1] *env 0x7ffe743b12f6 140730848449270 SESSION_MANAGER=local/fhy1190201816-virtual-mach env[2] *env 0x7ffe743b1376 140730848449398 QT_ACCESSIBILITY=1 env[3] *env 0x7ffe743b1389 140730848449417 COLORTERM=truecolor env[4] *env 0x7ffe743b139d 140730848449437 XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg env[5] *env 0x7ffe743b13ca 140730848449482

	<code>XDG_MENU_PREFIX=gnome-</code>
	<code>env[6] *env 0x7ffe743b13e1 140730848449505</code>
	<code>GNOME_DESKTOP_SESSION_ID=this-is-deprecated</code>
	<code>env[7] *env 0x7ffe743b140d 140730848449549</code>
	<code>GTK_IM_MODULE=fcitx</code>
	<code>env[8] *env 0x7ffe743b1421 140730848449569</code>
	<code>LANGUAGE=zh_CN:zh:en_US:en</code>
	<code>env[9] *env 0x7ffe743b143c 140730848449596</code>
	<code>QT4_IM_MODULE=fcitx</code>
	<code>env[10] *env 0x7ffe743b1450 140730848449616</code>
	<code>LC_ADDRESS=zh_CN.UTF-8</code>
	<code>env[11] *env 0x7ffe743b1467 140730848449639</code>
	<code>GNOME_SHELL_SESSION_MODE=ubuntu</code>
	<code>env[12] *env 0x7ffe743b1487 140730848449671</code>
	<code>LC_NAME=zh_CN.UTF-8</code>
	<code>env[13] *env 0x7ffe743b149b 140730848449691</code>
	<code>SSH_AUTH_SOCK=/run/user/1000/keyring/ssh</code>
	<code>env[14] *env 0x7ffe743b14c4 140730848449732</code>
	<code>XMODIFIERS=@im=fcitx</code>
	<code>env[15] *env 0x7ffe743b14d9 140730848449753</code>
	<code>DESKTOP_SESSION=ubuntu</code>

	<p>env[16] *env 0x7ffe743b14f0 140730848449776 LC_MONETARY=zh_CN.UTF-8</p> <p>env[17] *env 0x7ffe743b1508 140730848449800 SSH_AGENT_PID=1266</p> <p>env[18] *env 0x7ffe743b151b 140730848449819 GTK_MODULES=gail:atk-bridge</p> <p>env[19] *env 0x7ffe743b1537 140730848449847 PWD=/home/fhy-1190201816/桌面/HITICS</p> <p>env[20] *env 0x7ffe743b155e 140730848449886 XDG_SESSION_DESKTOP=ubuntu</p> <p>env[21] *env 0x7ffe743b1579 140730848449913 LOGNAME=fhy-1190201816</p> <p>env[22] *env 0x7ffe743b1590 140730848449936 XDG_SESSION_TYPE=x11</p> <p>env[23] *env 0x7ffe743b15a5 140730848449957 GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:</p> <p>env[24] *env 0x7ffe743b15d9 140730848450009 XAUTHORITY=/run/user/1000/gdm/Xauthority</p> <p>env[25] *env 0x7ffe743b1602 140730848450050 WINDOWPATH=2</p> <p>env[26] *env 0x7ffe743b160f 140730848450063</p>
--	--



	HOME=/home/fhy-1190201816
env[27]	*env 0x7ffe743b1629 140730848450089
	USERNAME=fhy-1190201816
env[28]	*env 0x7ffe743b1641 140730848450113
	IM_CONFIG_PHASE=1
env[29]	*env 0x7ffe743b1653 140730848450131
	LC_PAPER=zh_CN.UTF-8
env[30]	*env 0x7ffe743b1668 140730848450152
	LANG=zh_CN.UTF-8
env[31]	*env 0x7ffe743b1679 140730848450169
	LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:00;36:
env[32]	*env 0x7ffe743b1c68 140730848451688
	XDG_CURRENT_DESKTOP=ubuntu:GNOME
env[33]	*env 0x7ffe743b1c89 140730848451721
	VTE_VERSION=6200
env[34]	*env 0x7ffe743b1c9a 140730848451738

	<p>G_ENABLE_DIAGNOSTIC=0</p> <p>env[35] *env 0x7ffe743b1cb0 140730848451760</p> <p>GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen</p> <p>env[36] *env 0x7ffe743b1d06 140730848451846</p> <p>INVOCATION_ID=aae393fab2254cfcb3df4476db66eb78</p> <p>env[37] *env 0x7ffe743b1d35 140730848451893</p> <p>MANAGERPID=917</p> <p>env[38] *env 0x7ffe743b1d44 140730848451908</p> <p>CLUTTER_IM_MODULE=fcitx</p> <p>env[39] *env 0x7ffe743b1d5c 140730848451932</p> <p>LESSCLOSE=/usr/bin/lesspipe %s %s</p> <p>env[40] *env 0x7ffe743b1d7e 140730848451966</p> <p>XDG_SESSION_CLASS=user</p> <p>env[41] *env 0x7ffe743b1d95 140730848451989</p> <p>TERM=xterm-256color</p> <p>env[42] *env 0x7ffe743b1da9 140730848452009</p> <p>LC_IDENTIFICATION=zh_CN.UTF-8</p> <p>env[43] *env 0x7ffe743b1dc7 140730848452039</p> <p>LESSOPEN=  /usr/bin/lesspipe %s</p> <p>env[44] *env 0x7ffe743b1de7 140730848452071</p> <p>USER=fhy-1190201816</p>
--	--

env[45]	*env	0x7ffe743b1dfb	140730848452091
GNOME_TERMINAL_SERVICE=:1.102			
env[46]	*env	0x7ffe743b1e19	140730848452121
DISPLAY=:0			
env[47]	*env	0x7ffe743b1e24	140730848452132
SHLVL=1			
env[48]	*env	0x7ffe743b1e2c	140730848452140
LC_TELEPHONE=zh_CN.UTF-8			
env[49]	*env	0x7ffe743b1e45	140730848452165
QT_IM_MODULE=fcitx			
env[50]	*env	0x7ffe743b1e58	140730848452184
LC_MEASUREMENT=zh_CN.UTF-8			
env[51]	*env	0x7ffe743b1e73	140730848452211
PAPERSIZE=a4			
env[52]	*env	0x7ffe743b1e80	140730848452224
XDG_RUNTIME_DIR=/run/user/1000			
env[53]	*env	0x7ffe743b1e9f	140730848452255
LC_TIME=zh_CN.UTF-8			
env[54]	*env	0x7ffe743b1eb3	140730848452275
JOURNAL_STREAM=8:78940			
env[55]	*env	0x7ffe743b1eca	140730848452298

	XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share env[56] *env 0x7ffe743b1f1f 140730848452383 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/u env[57] *env 0x7ffe743b1f87 140730848452487 GDMSESSION=ubuntu env[58] *env 0x7ffe743b1f99 140730848452505 DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/100 env[59] *env 0x7ffe743b1fcf 140730848452559 LC_NUMERIC=zh_CN.UTF-8 env[60] *env 0x7ffe743b1fe6 140730848452582 _=./a.out gint0 0x41404140 1094730048 glong 0x404088 4210824 cstr 0x4040a0 4210848 pstr 0x402020 4202528 gc 0x40204c 4202572 cc 0x402060 4202592 local int 0x7ffe743aff8c 140730848444300 local int 10x7ffe743aff90 140730848444304 local static int 0 0x41404144 1094730052 local static int 1 0x404110 4210960
--	--

	local astr 0x7ffe743affd0 140730848444368
	local pstr 0x4020d0 4202704
	argc 0x7ffe743aff7c 140730848444284
	argv 0x7ffe743b04c8 140730848445640
	argv[0] 7ffe743b12c6
	argv[1] 7ffe743b12ce
	argv[2] 7ffe743b12d1
	argv[3] 7ffe743b12dc
	argv[0] 0x7ffe743b12c6 140730848449222
	./a.out
	argv[1] 0x7ffe743b12ce 140730848449230
	-u
	argv[2] 0x7ffe743b12d1 140730848449233
	1190201816
	argv[3] 0x7ffe743b12dc 140730848449244
	樊红雨

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 执行前:

```
<_init>:
<.plt>
<puts@plt>
<__stack_chk_fail@plt>
```

<\_\_printf\_chk@plt>  
<free@plt>  
<malloc@plt>  
<\_\_cxa\_finalize@plt>  
<\_start>  
<deregister\_tm\_clones>  
<register\_tm\_clones>  
<\_\_do\_global\_ctors\_aux>  
<frame\_dummy>  
<useless>  
<show\_pointer>

main 执行后:

<main>  
<\_\_libc\_csu\_init>  
<\_\_libc\_csu\_fini>  
<\_fini>

## 第3章 各阶段的原理与方法

每阶段 40 分，phases.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：

```
fhy-1190201816@fhy1190201816-virtual-machine:~/桌面/HITICS/linklab-1190201816$ gcc -m32 -no-pie -o linkbomb main.o phase1.o
fhy-1190201816@fhy1190201816-virtual-machine:~/桌面/HITICS/linklab-1190201816$ ./linkbomb
1190201816
fhy-1190201816@fhy1190201816-virtual-machine:~/桌面/HITICS/linklab-1190201816$
```

分析与设计的过程：

1. 使用 `readelf -a phase1.o` 命令查看 elf 文件的内容，发现字符串输出的其实地址.data 节中偏移量为 32。

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	00001e	00	AX	0	0	1
[ 2]	.rel.text	REL	00000000	0002bc	000010	08	I	11	1	4
[ 3]	.data	PROGBITS	00000000	000060	0000e4	00	WA	0	0	32
[ 4]	.rel.data	REL	00000000	0002cc	000008	08	I	11	3	4
[ 5]	.bss	NOBITS	00000000	000144	000000	00	WA	0	0	1
[ 6]	.comment	PROGBITS	00000000	000144	00002d	01	MS	0	0	1
[ 7]	.note.GNU-stack	PROGBITS	00000000	000171	000000	00		0	0	1
[ 8]	.note.gnu.pr[...]	NOTE	00000000	000174	00001c	00	A	0	0	4
[ 9]	.eh_frame	PROGBITS	00000000	000190	000038	00	A	0	0	4
[10]	.rel.eh_frame	REL	00000000	0002d4	000008	08	I	11	9	4
[11]	.symtab	SYMTAB	00000000	0001c8	0000d0	10		12	10	4
[12]	.strtab	STRTAB	00000000	000298	000021	00		0	0	1
[13]	.shstrtab	STRTAB	00000000	0002dc	00006e	00		0	0	1

Key to Flags:

2. 使用命令 `gcc -m32 -no-pie -o linkbomb main.o phase1.o`，将 main.o 和 phase1.o 链接成 linkbomb.o，运行 linkbomb 程序，看到原本该程序输出的字符串：

```
fhy-1190201816@fhy1190201816-virtual-machine:~/桌面/HITICS/linklab-1190201816$ ./linkbomb
SHEqv3T yJnLZJyYDvBts0G5WMKLE00K9S8HwTIbQSyQn7iQc6VW6Zc25DNGpmKvRhTDNlIUIjgrXzUMLTw8qz56EiBTLEi2Kaq3pw4v4hoHAsdcD pVK
```

3. 使用 hexedit，进入 phase1.o，将原本输出的字符串的十六进制修改为学号 1190201816 的十六进制。通过观察右边字符串，将原本的输出字符串替换，





观察 VqfjiByY 函数，在执行 strcmp 之前，将两个参数入栈，其中一个参数的地址为：0x804a07c。

通过 edb 查看该内存中存储的变量：

```
leal 0x804a07c, %eax | ASCII "1190201816"
```

发现 0x804a07c 存储的变量为我的学号。

那么另一个变量即为 VqfjiByY 函数的参数。

```
0804922e <do_phase>:
804922e: f3 0f 1e fb          endbr32
8049232: 55                  push %ebp
8049233: 89 e5               mov %esp,%ebp
```

do\_phase 剩余代码为 nop，留给我们进行补充。

- 根据 1 中的分析，我们知道了，我们需要在 do\_phase 中插入汇编代码，将我的学号传入 VqfjiByY 函数中，并且调用 VqfjiByY 函数。

所以记下 VqfjiByY 函数的地址：0x80491fa。

- 编写汇编代码：

```
1 lea 0x804a07c,%eax
2 lea 0x080491fa,%ecx
3 push %eax
4 call *%ecx
5 pop %eax
```

首先，将 0x804a07c 即存储学号的地址传入寄存器 %eax 中，将 VqfjiByY 函数的首地址 0x80491fa 传入寄存器 %ecx 中，将寄存器 %eax 入栈，即将我的学号作为参数入栈，调用函数 VqfjiByY，在调用完函数 VqfjiByY 后，将 %eax pop 出栈，使栈帧恢复原状，防止发生段错误。

- 查看汇编代码的十六进制：

使用 gcc -m32 -c 命令将汇编代码编译为可重定位文件，利用 objdump -d 命令查看汇编代码的十六进制：

```
00000000 <.text>:
```

0804:9250	8d 55 e9	leal -0x17(%ebp), %edx
0804:9253	8b 45 e4	movl -0x1c(%ebp), %eax
0804:9256	01 d0	addl %edx, %eax
0804:9258	0f b6 00	movzbl 0(%eax), %eax
0804:925b	0f b6 c0	movzbl %al, %eax
0804:925e	0f b6 80 40 c0 04 08	movzbl 0x804c040(%eax), %eax
0804:9265	0f be c0	movsbl %al, %eax
0804:9268	83 ec 0c	subl \$0xc, %esp
0804:926b	50	pushl %eax
0804:926c	e8 3f fe ff ff	calll 0x80490b0
0804:9271	83 c4 10	addl \$0x10, %esp
0804:9274	83 45 e4 01	addl \$1, -0x1c(%ebp)
0804:9278	8b 45 e4	movl -0x1c(%ebp), %eax
0804:927b	83 f8 09	cmpl \$9, %eax
0804:927e	76 d0	jbe 0x8049250

发现在进入循环后，寄存器中保存的一个字符串 `jouvaxrhqw` 在被一个字符一个字符读取，并且这个字符串的长度刚好与学号长度相同，这与 PPT 中的 cookie 的操作完全相同，所以猜测该字符串即为 cookie 串。

2. 我们现在需要查看全局变量 `char PHASE3_CODEBOOK[256]` 的名称。

利用 `readelf -s` 命令查看 `phase3.o` 的符号表：

```

fhy-1190201816@fhy1190201816-virtual-machine:~/桌面/HITICS/linklab-1190201816$ r
readelf -s phase3.o

Symbol table '.symtab' contains 14 entries:
Num:  Value      Size  Type  Bind  Vis  Ndx  Name
 0: 00000000      0 NOTYPE LOCAL DEFAULT UND
 1: 00000000      0 FILE  LOCAL DEFAULT ABS phase3.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 7
 6: 00000000      0 SECTION LOCAL DEFAULT 8
 7: 00000000      0 SECTION LOCAL DEFAULT 9
 8: 00000000      0 SECTION LOCAL DEFAULT 6
 9: 00000020    256 OBJECT GLOBAL DEFAULT COM FJSVvWtATw
10: 00000000    135 FUNC  GLOBAL DEFAULT 1 do_phase
11: 00000000      0 NOTYPE GLOBAL DEFAULT UND putchar
12: 00000000      0 NOTYPE GLOBAL DEFAULT UND __stack_chk_fail
13: 00000000      4 OBJECT GLOBAL DEFAULT 3 phase

```

发现只有一个长度为 256 并且类型为 COM 的变量，名称为 `FJSVvWtATw`。

可以利用强弱符号的性质，我们在链接时将该数组进行初始化，内容为根据 cookie 计算出的学号的排列。这样就可以使 `do_phase` 输出我们的学号。

3. 建立对应关系：

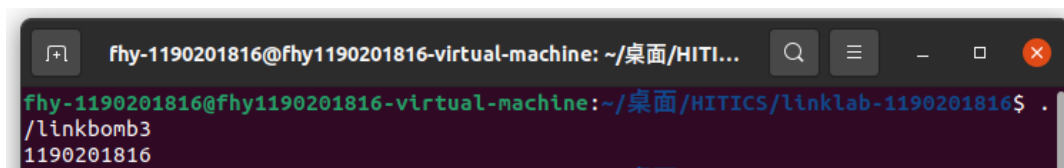
Cookie 串	j	o	u	v	a	x	r	h	q	w
ASCII 码	106	111	117	118	97	120	114	104	113	119
对应学号	1	1	9	0	2	0	1	8	1	6

根据对应规则，对应学号在长度为 256 的字符数组中的下标即为对应字母的 ASCII 码。可以写一个 C 程序来生成这个数组：



使用 `gcc -m32 -no-pie -o linkbomb3 main.o phase3.o phase3_patch.o` 进行链接操作，生成可执行文件 `linkbomb3`。

#### 4. 运行结果：

A terminal window with a dark background. The title bar shows 'fhy-1190201816@fhy1190201816-virtual-machine: ~/桌面/HITl...'. The prompt is 'fhy-1190201816@fhy1190201816-virtual-machine:~/桌面/HITICS/linklab-1190201816\$'. The user has entered './linkbomb3' and the output is '1190201816'.

### 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

学会了 hexedit 工具的使用；  
学会了将多个.o 文件链接在一起运行；  
学会了 readelf 查看 elf 头文件。

### 4.2 请给出对本次实验内容的建议

希望老师可以将第二阶段进行优化，因为我感觉第二阶段与上个实验类似，都可以利用修改栈帧的方式使程序输出学号。

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>.