

哈尔滨工业大学

实验报告

实 验（五）

题 目 LinkLab

链接

专 业 计算学部

学 号 1190200717

班 级 1903008

学 生 梁浩

指 导 教 师 吴锐

实 验 地 点 G709

实 验 日 期 2021/5/24

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.....	- 3 -
1.2.2 软件环境.....	- 3 -
1.2.3 开发工具.....	- 3 -
1.3 实验预习.....	- 3 -
第 2 章 实验预习.....	- 5 -
2.1 ELF 文件格式解读.....	- 5 -
2.2 程序的内存映像结构.....	- 5 -
2.3 程序中符号的位置分析.....	- 6 -
2.4 程序运行过程分析.....	- 9 -
第 3 章 各阶段的原理与方法.....	- 10 -
3.1 阶段 1 的分析.....	- 10 -
3.2 阶段 2 的分析.....	- 11 -
3.3 阶段 3 的分析.....	- 12 -
3.4 阶段 4 的分析.....	- 13 -
3.5 阶段 5 的分析.....	- 14 -
第 4 章 总结.....	- 15 -
4.1 请总结本次实验的收获.....	- 15 -
4.2 请给出对本次实验内容的建议.....	- 15 -
参考文献.....	- 16 -

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

处理器: Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40GHz

已安装的内存(RAM): 8.00GB(7.81GB 可用)

系统类型: 64 位操作系统, 基于 x64 的处理器

1.2.2 软件环境

Windows 10 家庭中文版; VirtualBox 6.1; Ubuntu 20.04

1.2.3 开发工具

GDB, OBJDUMP, Hexedit

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 ELF 文件格式解读

请按顺序写出 ELF 格式的可执行目标文件的各类信息（5 分）

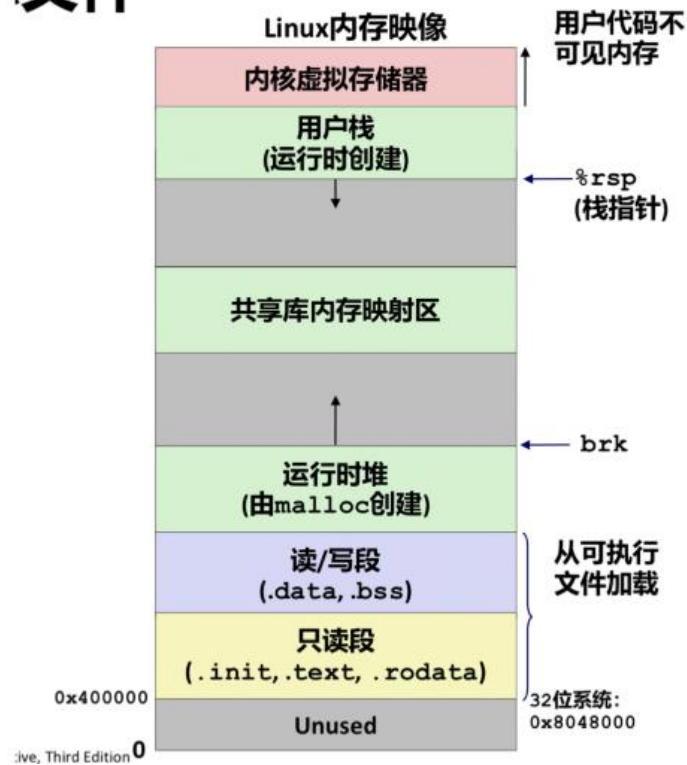
ELF 头	包括 16 字节标识信息、文件类型、机器类型、节头表的偏移、节头表的表项大小以及表项个数
程序头表	一个结构数组，描述可执行文件中的节与虚拟空间中的存储段之间的映射关系
.init 节	定义 <code>_init</code> 函数，用来进行可执行目标文件开始执行时的初始化工作
.text 节	编译后的代码部分
.rodata 节	只读数据，如 <code>printf</code> 格式串、 <code>switch</code> 跳转表等
.data 节	已初始化的全局变量
.bss 节	未初始化全局变量，仅是占位符，不占据任何实际磁盘空间
.symtab 节	存放函数和全局变量（符号表）信息，不包括局部变量
.debug	调试信息
.line	本节也是一个用于调试的节,它包含那些调试符号的行号,为程序指令码与源文件的行号建立起联系
.strtab	用于存放字符串,主要是那些符号表项的名字
节头表（可重定位目标文件）	描述每个节的节名、在文件中的偏移、大小、访问属性、对齐方式等

2.2 程序的内存映像结构

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）

Linux下X64内存映像

文件



2.3 程序中符号的位置分析

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（5 分）

所属区	各符号的地址、空间
只 读 代 码 段 (.init, .text, .rodata)	<pre> exit 0x7f2019066bc0 139775835532224 printf 0x7f2019081e10 139775835643408 malloc 0x7f20190ba260 139775835873888 free 0x7f20190ba850 139775835875408 </pre>
读 写 段 (.data, .bss)	<pre> show_pointer 0x55e72102a199 94451179626905 useless 0x55e72102a1d0 94451179626960 main 0x55e72102a1df 94451179626975 big array 0x55e76102d040 94452253380672 huge array 0x55e72102d040 94451179638848 global 0x55e72102d02c 94451179638828 </pre>

运行时堆（由 malloc 创建）	<p>p1 0x7f200901c010 139775566790672</p> <p>p2 0x55e762a756b0 94452280940208</p> <p>p3 0x7f2008ffb010 139775566655504</p> <p>p4 0x7f1fc8ffa010 139774492909584</p> <p>p5 0x7f1f48ff9010 139772345421840</p>
用户栈（运行时创建）	<p>argc 0x7ffe16892c9c 140729276509340</p> <p>argv 0x7ffe16892dd8 140729276509656</p> <p>argv[0] 7ffe168932d4</p> <p>argv[1] 7ffe168932e2</p> <p>argv[2] 7ffe168932e5</p> <p>argv[3] 7ffe168932f0</p> <p>argv[0] 0x7ffe168932d4 140729276510932</p> <p>./linkaddress</p> <p>argv[1] 0x7ffe168932e2 140729276510946</p> <p>-u</p> <p>argv[2] 0x7ffe168932e5 140729276510949</p> <p>1190200717</p> <p>argv[3] 0x7ffe168932f0 140729276510960</p> <p>梁浩</p>

	<pre> env 0x7ffe16892e00 140729276509696 env[0] *env 0x7ffe168932f7 140729276510967 SHELL=/bin/bash env[1] *env 0x7ffe16893307 140729276510983 SESSION_MANAGER=local/Graham:@/tmp/.ICE-unix/1549,unix/Graham:/tmp/.ICE-unix/1549 env[2] *env 0x7ffe16893359 140729276511065 QT_ACCESSIBILITY=1 env[3] *env 0x7ffe1689336c 140729276511084 COLORTERM=truecolor env[4] *env 0x7ffe16893380 140729276511104 XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg env[5] *env 0x7ffe168933ad 140729276511149 XDG_MENU_PREFIX=gnome- env[6] *env 0x7ffe168933c4 140729276511172 GNOME_DESKTOP_SESSION_ID=this-is-deprecated env[7] *env 0x7ffe168933f0 140729276511216 LANGUAGE=zh_CN:zh env[8] *env 0x7ffe16893402 140729276511234 LC_ADDRESS=zh_CN.UTF-8 env[9] *env 0x7ffe16893419 140729276511257 GNOME_SHELL_SESSION_MODE=ubuntu env[10] *env 0x7ffe16893439 140729276511289 LC_NAME=zh_CN.UTF-8 env[11] *env 0x7ffe1689344d 140729276511309 SSH_AUTH_SOCK=/run/user/1001/keyring/ssh env[12] *env 0x7ffe16893476 140729276511350 XMODIFIERS=@im=ibus env[13] *env 0x7ffe1689348a 140729276511370 DESKTOP_SESSION=ubuntu env[14] *env 0x7ffe168934a1 140729276511393 LC_MONETARY=zh_CN.UTF-8 env[15] *env 0x7ffe168934b9 140729276511417 SSH_AGENT_PID=1501 env[16] *env 0x7ffe168934cc 140729276511436 GTK_MODULES=gail:atk-bridge env[17] *env 0x7ffe168934e8 140729276511464 DBUS_STARTER_BUS_TYPE=session env[18] *env 0x7ffe16893506 140729276511494 PWD=/home/1190200717lh/labfive env[19] *env 0x7ffe16893525 140729276511525 LOGNAME=1190200717lh env[20] *env 0x7ffe1689353a 140729276511546 XDG_SESSION_DESKTOP=ubuntu env[21] *env 0x7ffe16893555 140729276511573 XDG_SESSION_TYPE=x11 env[22] *env 0x7ffe1689356a 140729276511594 GPG_AGENT_INFO=/run/user/1001/gnupg/S.gpg-agent:0:1 env[23] *env 0x7ffe1689359e 140729276511646 XAUTHORITY=/run/user/1001/gdm/Xauthority env[24] *env 0x7ffe168935c7 140729276511687 WINDOWPATH=2 env[25] *env 0x7ffe168935d4 140729276511700 HOME=/home/1190200717lh env[26] *env 0x7ffe168935ec 140729276511724 USER=1190200717lh env[27] *env 0x7ffe16893602 140729276511746 IN_CONFIG_PHASE=1 env[28] *env 0x7ffe16893614 140729276511764 LC_PAPER=zh_CN.UTF-8 env[29] *env 0x7ffe16893629 140729276511785 LANG=zh_CN.UTF-8 env[30] *env 0x7ffe1689363a 140729276511802 LC_CTYPE=zh_CN.UTF-8 env[31] *env 0x7ffe1689364d 140729276511825 LC_NUMERIC=zh_CN.UTF-8 env[32] *env 0x7ffe1689365d 140729276511848 LC_TIME=zh_CN.UTF-8 env[33] *env 0x7ffe1689366d 140729276511871 LC_MESSAGES=zh_CN.UTF-8 env[34] *env 0x7ffe1689367d 140729276511894 LC_IDENTIFICATION=zh_CN.UTF-8 env[35] *env 0x7ffe1689368d 140729276511917 LC_MONETARY=zh_CN.UTF-8 env[36] *env 0x7ffe1689369d 140729276511940 LC_NUMERIC=zh_CN.UTF-8 env[37] *env 0x7ffe168936ad 140729276511963 LC_TIME=zh_CN.UTF-8 env[38] *env 0x7ffe168936bd 140729276511986 LC_MESSAGES=zh_CN.UTF-8 env[39] *env 0x7ffe168936cd 140729276512009 LC_IDENTIFICATION=zh_CN.UTF-8 env[40] *env 0x7ffe168936dd 140729276512032 LC_MONETARY=zh_CN.UTF-8 env[41] *env 0x7ffe168936ed 140729276512055 LC_TIME=zh_CN.UTF-8 env[42] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[43] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[44] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[45] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[46] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[47] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[48] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[49] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[50] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[51] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[52] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[53] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[54] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[55] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[56] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[57] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[58] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[59] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[60] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[61] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[62] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[63] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[64] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[65] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[66] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[67] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[68] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[69] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[70] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[71] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[72] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[73] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[74] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[75] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[76] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[77] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[78] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[79] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[80] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[81] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[82] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[83] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[84] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[85] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[86] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[87] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[88] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[89] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[90] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[91] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[92] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[93] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[94] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[95] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 env[96] *env 0x7ffe168936fd 140729276512078 LC_MONETARY=zh_CN.UTF-8 env[97] *env 0x7ffe168936fd 140729276512078 LC_TIME=zh_CN.UTF-8 env[98] *env 0x7ffe168936fd 140729276512078 LC_MESSAGES=zh_CN.UTF-8 env[99] *env 0x7ffe168936fd 140729276512078 LC_IDENTIFICATION=zh_CN.UTF-8 </pre>
--	---

	<pre> env[40] *env 0x7ffe16893d4e 140729276513614 LESSOPEN= /usr/bin/lesspipe %s env[41] *env 0x7ffe16893d6e 140729276513646 USER=1190200717lh env[42] *env 0x7ffe16893d80 140729276513664 GNOME_TERMINAL_SERVICE=:1.80 env[43] *env 0x7ffe16893d9d 140729276513693 DISPLAY=:0 env[44] *env 0x7ffe16893da8 140729276513704 SHLVL=1 env[45] *env 0x7ffe16893db0 140729276513712 LC_TELEPHONE=zh_CN.UTF-8 env[46] *env 0x7ffe16893dc9 140729276513737 QT_IM_MODULE=ibus env[47] *env 0x7ffe16893ddb 140729276513755 LC_MEASUREMENT=zh_CN.UTF-8 env[48] *env 0x7ffe16893df6 140729276513782 DBUS_STARTER_ADDRESS=unix:path=/run/user/1001/bus,guid=2a863e8c65fda94650ee147060acec2f env[49] *env 0x7ffe16893e4e 140729276513870 XDG_RUNTIME_DIR=/run/user/1001 env[50] *env 0x7ffe16893e6d 140729276513901 LC_TIME=zh_CN.UTF-8 env[51] *env 0x7ffe16893e81 140729276513921 JOURNAL_STREAM=8:33612 env[52] *env 0x7ffe16893e98 140729276513944 XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd/desktop env[53] *env 0x7ffe16893eed 140729276514029 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin env[54] *env 0x7ffe16893f55 140729276514133 GDMSESSION=ubuntu env[55] *env 0x7ffe16893f67 140729276514151 DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1001/bus,guid=2a863e8c65fda94650ee147060acec2f env[56] *env 0x7ffe16893fc3 140729276514243 LC_NUMERIC=zh_CN.UTF-8 env[57] *env 0x7ffe16893fda 140729276514266 _=./linkaddress local 0x7ffe16892ca0 140729276509344 </pre>
--	---

2.4 程序运行过程分析

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(使用 gcc 与 objdump/GDB/EDB) (5 分)

main 执行前:

```

<_init>
<.plt>
<free@plt>
<printf@plt>
<malloc@plt>
<__cxa_finalize@plt>
<puts@plt>
<__stack_chk_fail@plt>
<_start>
<deregister_tm_clones>
<register_tm_clones>
<__do_global_ctors_aux>
<frame_dummy>
<show_pointer>

```

<useless>main 执行后:

```

<__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：

```
1190200717lh@Graham: ~/labfive/linklab-1190200717
1190200717lh@Graham:~/labfive/linklab-1190200717$ gcc -m32 -o linkbomb1 main.o phase1.o
1190200717lh@Graham:~/labfive/linklab-1190200717$ ./linkbomb1
1190200717
1190200717lh@Graham:~/labfive/linklab-1190200717$
```

分析与设计的过程：

1) 未修改前

先试着用未修改过的 phase1.o 和 main.o 链接得到 linkbomb1，执行后输入如下图所示的字符串。

```
1190200717lh@Graham: ~/labfive/linklab-1190200717
1190200717lh@Graham:~/labfive/linklab-1190200717$ gcc -m32 -o linkbomb1 main.o phase1.o
1190200717lh@Graham:~/labfive/linklab-1190200717$ ./linkbomb1
cE      fYzMaz5c8iIEuqihLS8lUmpFCCX6rXcsDBWT9sRmi3dfEQ  CewCgu9NAMA 8i1U5vxhs
1190200717lh@Graham:~/labfive/linklab-1190200717$
```

2) 寻找字符串存储的位置

方法一：

直接用 hexedit 打开 phase1.o，根据左右两边的对照，可以寻找到字符串的位置，如下图所示：

```
77 43 43 47 6F 64 09 39 4F 50 35 6A 65 62 55 32 63 45 09 66 59 7A 4D 61 7A 35 63 38 69 49 45 75 71 69 68 4C 53 38 6C 55 wCCGod.90P5jebU2ce.fYzMaz5c8iIEuqihLS8lU
60 70 46 43 43 58 36 72 58 63 73 44 42 57 54 39 73 52 6D 69 33 64 66 45 51 09 43 65 77 43 67 75 39 4E 41 4D 61 20 38 69 mpFCCX6rXcsDBWT9sRmi3dfEQ.CewCgu9NAMA 8i
31 55 35 76 78 68 73 00 00 00 00 00 47 43 43 3A 20 28 55 62 75 6E 74 75 20 39 2E 32 2E 31 2D 39 75 62 75 6E 74 75 32 1U5vxhs.....GCC: (Ubuntu 9.2.1-9ubuntu2
```

利用 `readelf -a phase1.o`, 得知 `.data` 区对于 `phase1.o` 的偏移量是 `0x60`

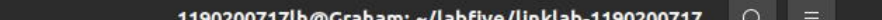
再通过 objdump 查看 phase1.o 的具体实现过程:

call 14 <do phase+0x14>是调用了 puts 函数，且将 0x28 作为参数传了进去。

因此字符串所在的位置应在 $0x28 + 0x60 = 0x88$ 处。经查证，确实如此。

3) 将原本的字符串修改为学号

多余的地方用 00 填充。



```
1190200717lh@Graham: ~/labfive/linklab-1190200717
1190200717lh@Graham:~/labfive/linklab-1190200717$ gcc -m32 -o linkbomb1 main.o phase1.o
1190200717lh@Graham:~/labfive/linklab-1190200717$ ./linkbomb1
1190200717
1190200717lh@Graham:~/labfive/linklab-1190200717$
```

3.2 阶段 2 的分析

分析与设计的过程:

3.3 阶段 3 的分析

程序运行结果截图：



```
1190200717lh@Graham: ~/labfive/linklab-1190200717
1190200717lh@Graham:~/labfive/linklab-1190200717$ gcc -m32 -o linkbomb3 main.o phase3.o phase3_patch.o
1190200717lh@Graham:~/labfive/linklab-1190200717$ ./linkbomb3
1190200717
1190200717lh@Graham:~/labfive/linklab-1190200717$
```

分析与设计的过程：

1) 根据 PPT 提示分析目标

■ phase3.c程序框架

```
char PHASE3_CODEBOOK[256];
void do_phase(){
    const char char cookie[] = PHASE3_COOKIE;
    for( int i=0; i<sizeof(cookie)-1; i++ )
        printf( "%c", PHASE3_CODEBOOK[ (unsigned char)(cookie[i]) ] );
    printf( "\n" );
}
```

从 PPT 中给出的 do_phase()函数可知,此题设计到两个字符串数组,分别是 cookie[] 和 PHASE3_CODEBOOK[256]。具体来说,遍历 cookie[]中的每个字符,将这些字符的 ASCII 值作为下标,输出 PHASE3_CODEBOOK[256]对应下标下的字符。因此,我们应该弄明白这两个数组的内容。

2) 寻找 PHASE3_CODEBOOK[256]

3)通过符号表,发现该数组为一未初始化变量(类型为COM,长度为256字节)

根据提示,通过 readelf -a phase3.o 指令,在.symtab 节我们找到了未初始化的数组,数组名为 PnpwxWMXVK,因此我们应该在 phase3_patch.c 中定义同名的数组,从而在链接时替换对原数组的引用。

```
9: 00000020 256 OBJECT GLOBAL DEFAULT COM PnpwxWMXVK
```

3) 寻找 cookie[]

利用 gdb 调试 linkbomb3,运行到下图所示的指令,查看%ebp-0x17 存的值,发现

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

- 1、学会了 hexedit 的使用
- 2、练习了将多个.o 文件链接在一起生成可执行文件
- 3、熟悉了 ELF 格式的可执行目标文件的各类信息
- 4、体会了链接过程中用强符号取代同名弱符号的过程

4.2 请给出对本次实验内容的建议

无

参考文献

- [1] 深入理解计算机系统