

# 哈尔滨工业大学

# 实验报告

## 实 验（五）

题 目 LinkLab

链接

专 业 计算机系

学 号 1190201421

班 级 1936603

学 生 张瑞

指 导 教 师 刘宏伟

实 验 地 点 G709

实 验 日 期 2021 年 5 月 20 日

## 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>- 5 -</b>
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	- 5 -
2.4 程序运行过程分析	- 9 -
<b>第 3 章 各阶段的原理与方法</b>	<b>- 11 -</b>
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	- 15 -
3.4 阶段 4 的分析	- 15 -
3.5 阶段 5 的分析	- 16 -
<b>第 4 章 总结</b>	<b>- 17 -</b>
4.1 请总结本次实验的收获	- 17 -
4.2 请给出对本次实验内容的建议	- 17 -
<b>参考文献</b>	<b>- 18 -</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 内存映像结构, 标出其所属各区。

```
gcc -m64 -o LinkAddress linkaddress.c
```

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

ELF 头：描述文件总体格式，还包括程序入口点

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，以及所有被初始化为 0 的全局或静态变量

.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

.strtab：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 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

(Unused)

只读代码段(`.init`, `.text`, `.rodata`)

读/写段(`.data`, `.bss`)

运行时堆(由 `malloc` 创建)

共享库内存映射区

用户栈(运行时创建)

(内核内存(对用户代码不可见的内存))

### 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，并按照 Linux 下 X64 内存映像标出其所属内存区段

所属区	各符号的地址、空间（地址从小到大）
只读代码段 （.init,.text,.redat a）	exit 0x7f2f2d5b9bc0 139840601168832 printf 0x7f2f2d5d4e10 139840601280016 malloc 0x7f2f2d60d260 139840601510496 free 0x7f2f2d60d850 139840601512016
读 写 段 （.data,.bss）	show_pointer 0x55d23fb6d199 94361500438937 useless 0x55d23fb6d1d0 94361500438992 main 0x55d23fb6d1df 94361500439007 global 0x55d23fb7002c 94361500450860 huge array 0x55d23fb70040 94361500450880 big array 0x55d27fb70040 94362574192704
运 行 时 堆 （ 由 malloc 创建）	p1 0x7f2f1d56f010 139840332427280 p2 0x55d2815e06b094362601916080 p3 0x7f2f1d54e010 139840332292112 p4 0x7f2edd54d010 139839258546192 p5 0x7f2e5d54c010 139837111058448
用户栈（运行时创 建）	argc0x7ffd58fff20c 140726096622092 local 0x7ffd58fff210 140726096622096 argv 0x7ffd58fff348 140726096622408 argv[0] 7ffd590012b8 argv[1] 7ffd590012c6 argv[2] 7ffd590012c9 argv[3] 7ffd590012d4 argv[0] 0x7ffd590012b8 140726096630456 ./LinkAddress argv[1] 0x7ffd590012c6 140726096630470 -u argv[2] 0x7ffd590012c9 140726096630473 1190201421 argv[3] 0x7ffd590012d4 140726096630484 张瑞 env 0x7ffd58fff370 140726096622448 env[0] *env 0x7ffd590012db 140726096630491 SHELL=/bin/bash env[1] *env 0x7ffd590012eb 140726096630507 SESSION_MANAGER=local/ubuntu:@/tmp/.ICE-unix/1713,unix/ubuntu:/tmp/.ICE-unix/1713 env[2] *env 0x7ffd5900133d 140726096630589 QT_ACCESSIBILITY=1 env[3] *env 0x7ffd59001350 140726096630608 COLORTERM=truecolor env[4] *env 0x7ffd59001364 140726096630628 XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg env[5] *env 0x7ffd59001391 140726096630673 XDG_MENU_PREFIX=gnome- env[6] *env 0x7ffd590013a8 140726096630696 GNOME_DESKTOP_SESSION_ID=this-is-deprecated

```

env[7] *env 0x7ffd590013d4 140726096630740
GTK_IM_MODULE=fcitx
env[8] *env 0x7ffd590013e8 140726096630760
LANGUAGE=zh_CN:en
env[9] *env 0x7ffd590013fa 140726096630778
QT4_IM_MODULE=fcitx
env[10] *env 0x7ffd5900140e 140726096630798
LC_ADDRESS=zh_CN.UTF-8
env[11] *env 0x7ffd59001425 140726096630821
GNOME_SHELL_SESSION_MODE=ubuntu
env[12] *env 0x7ffd59001445 140726096630853
LC_NAME=zh_CN.UTF-8
env[13] *env 0x7ffd59001459 140726096630873
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
env[14] *env 0x7ffd59001482 140726096630914
XMODIFIERS=@im=fcitx
env[15] *env 0x7ffd59001497 140726096630935
DESKTOP_SESSION=ubuntu
env[16] *env 0x7ffd590014ae 140726096630958
LC_MONETARY=zh_CN.UTF-8
env[17] *env 0x7ffd590014c6 140726096630982
SSH_AGENT_PID=1670
env[18] *env 0x7ffd590014d9 140726096631001
GTK_MODULES=gail:atk-bridge
env[19] *env 0x7ffd590014f5 140726096631029
DBUS_STARTER_BUS_TYPE=session
env[20] *env 0x7ffd59001513 140726096631059
PWD=/home/zr/shared
env[21] *env 0x7ffd59001527 140726096631079
LOGNAME=zr
env[22] *env 0x7ffd59001532 140726096631090
XDG_SESSION_DESKTOP=ubuntu
env[23] *env 0x7ffd5900154d 140726096631117
XDG_SESSION_TYPE=x11
env[24] *env 0x7ffd59001562 140726096631138
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
env[25] *env 0x7ffd59001596 140726096631190
XAUTHORITY=/run/user/1000/gdm/Xauthority
env[26] *env 0x7ffd590015bf 140726096631231
WINDOWPATH=2
env[27] *env 0x7ffd590015cc 140726096631244
HOME=/home/zr
env[28] *env 0x7ffd590015da 140726096631258
USERNAME=zr
env[29] *env 0x7ffd590015e6 140726096631270
IM_CONFIG_PHASE=1
env[30] *env 0x7ffd590015f8 140726096631288

```

```

LC_PAPER=zh_CN.UTF-8
env[31] *env 0x7ffd5900160d 140726096631309
LANG=zh_CN.UTF-8
env[32] *env 0x7ffd5900161e 140726096631326
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40
1;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=
*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=0
;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:
35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:
*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.o
env[33] *env 0x7ffd59001c00 140726096632832
XDG_CURRENT_DESKTOP=ubuntu:GNOME
env[34] *env 0x7ffd59001c21 140726096632865
VTE_VERSION=6003
env[35] *env 0x7ffd59001c32 140726096632882
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/6ce9dc96_0f7a_421
env[36] *env 0x7ffd59001c88 140726096632968
INVOCATION_ID=8aa5e77148fb44c38055ad6609f9ff5d
env[37] *env 0x7ffd59001cb7 140726096633015
MANAGERPID=1502
env[38] *env 0x7ffd59001cc7 140726096633031
CLUTTER_IM_MODULE=fcitx
env[39] *env 0x7ffd59001cdf 140726096633055
LESSCLOSE=/usr/bin/lesspipe %s %s
env[40] *env 0x7ffd59001d01 140726096633089
XDG_SESSION_CLASS=user
env[41] *env 0x7ffd59001d18 140726096633112
TERM=xterm-256color
env[42] *env 0x7ffd59001d2c 140726096633132
LC_IDENTIFICATION=zh_CN.UTF-8
env[43] *env 0x7ffd59001d4a 140726096633162
LESSOPEN=| /usr/bin/lesspipe %s
env[44] *env 0x7ffd59001d6a 140726096633194
USER=zr
env[45] *env 0x7ffd59001d72 140726096633202
GNOME_TERMINAL_SERVICE=:1.84
env[46] *env 0x7ffd59001d8f 140726096633231
DISPLAY=:0
env[47] *env 0x7ffd59001d9a 140726096633242
SHLVL=1
env[48] *env 0x7ffd59001da2 140726096633250
LC_TELEPHONE=zh_CN.UTF-8
env[49] *env 0x7ffd59001dbb 140726096633275
QT_IM_MODULE=fcitx
env[50] *env 0x7ffd59001dce 140726096633294
LC_MEASUREMENT=zh_CN.UTF-8
env[51] *env 0x7ffd59001de9 140726096633321

```



	<pre> DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=c7011480d7e4c env[52] *env 0x7ffd59001e41 140726096633409 PAPERSIZE=a4 env[53] *env 0x7ffd59001e4e 140726096633422 XDG_RUNTIME_DIR=/run/user/1000 env[54] *env 0x7ffd59001e6d 140726096633453 LC_TIME=zh_CN.UTF-8 env[55] *env 0x7ffd59001e81 140726096633473 JOURNAL_STREAM=8:48445 env[56] *env 0x7ffd59001e98 140726096633496 XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd/de env[57] *env 0x7ffd59001eed 140726096633581 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loca env[58] *env 0x7ffd59001f55 140726096633685 GDMSESSION=ubuntu env[59] *env 0x7ffd59001f67 140726096633703 DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=c7011480 env[60] *env 0x7ffd59001fc3 140726096633795 LC_NUMERIC=zh_CN.UTF-8 env[61] *env 0x7ffd59001fda 140726096633818 = ./LinkAddress </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)

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，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：

```

zr@ubuntu:~/shared/linklab-1190201421$ gcc -m32 -o linkbomb1 main.o phase1.o
zr@ubuntu:~/shared/linklab-1190201421$ ./linkbomb1
1190201421

```

分析与设计的过程：

用 `readelf -a phase1.o` 命令，发现.data 节偏移量为 0x60：

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	0002a4	000010	08	I 11	1	4	
[ 3]	.data	PROGBITS	00000000	000060	0000cc	00	WA	0	0	32
[ 4]	.rel.data	REL	00000000	0002b4	000008	08	I 11	3	4	
[ 5]	.bss	NOBITS	00000000	00012c	000000	00	WA	0	0	1
[ 6]	.comment	PROGBITS	00000000	00012c	00002d	01	MS	0	0	1
[ 7]	.note.GNU-stack	PROGBITS	00000000	000159	000000	00		0	0	1
[ 8]	.note.gnu.property	NOTE	00000000	00015c	00001c	00	A	0	0	4
[ 9]	.eh_frame	PROGBITS	00000000	000178	000038	00	A	0	0	4
[10]	.rel.eh_frame	REL	00000000	0002bc	000008	08	I 11	9	4	
[11]	.symtab	SYMTAB	00000000	0001b0	0000d0	10		12	10	4
[12]	.strtab	STRTAB	00000000	000280	000021	00		0	0	1
[13]	.shstrtab	STRTAB	00000000	0002c4	00006e	00		0	0	1

先用命令 `gcc -m32 -o linkbomb1 main.o phase1.o` 将 main.o 和 phase1.o 链接成 linkbomb1.o，然后运行 linkbomb1 程序，查看本来应该输出的字符串：

```

zr@ubuntu:~/shared/linklab-1190201421$ gcc -m32 -o linkbomb1 main.o phase1.o
zr@ubuntu:~/shared/linklab-1190201421$ ./linkbomb1
LHFzZ0AFatRdrJcXSDajswvzEGYcaB6wSHO 6NexnZaeaaQMUomjGwyXqsrEzsVudd0N InbpsBhLvACaUQ0
ME49 zi7o2ZEsdSOAOVtxIRVZviYn1r1FELEyiDxXj79TvzTj0krVxL WmUz8FR SfjOjc Ehmgq
ujzYff0nlkAHv1VhVefMvvSYsZ82rAEIP

```

利用 Winhex 查看 phase1.o，通过观察最右边的字符找到输出的字符串：

用学号 1190201421 对应的 `ascii` 码 31 31 39 30 32 30 31 34 32 31 替换它，再加上 00 作为字符串的结束：

然后再进行链接输出便可得到 1190201421:

## 阶段 2 的分析

### 分析与设计的过程:

- 12 -

```

00001231 <nJolafiK>:
    1231:    f3 0f 1e fb          endbr32
    1235:    55                  push    %ebp
    1236:    89 e5                mov     %esp,%ebp
    1238:    83 ec 08             sub     $0x8,%esp
    123b:    83 ec 08             sub     $0x8,%esp
    123e:    68 7c 20 00 00        push   $0x207c
    1243:    ff 75 08             pushl   0x8(%ebp)
    1246:    e8 fc ff ff ff        call    1247 <nJolafiK+0x16>
    124b:    83 c4 10             add     $0x10,%esp
    124e:    85 c0                test    %eax,%eax
    1250:    75 10                jne     1262 <nJolafiK+0x31>
    1252:    83 ec 0c             sub     $0xc,%esp
    1255:    ff 75 08             pushl   0x8(%ebp)
    1258:    e8 fc ff ff ff        call    1259 <nJolafiK+0x28>
    125d:    83 c4 10             add     $0x10,%esp
    1260:    eb 01                jmp     1263 <nJolafiK+0x32>
    1262:    90                  nop
    1263:    c9                  leave
    1264:    c3                  ret

00001265 <do_phase>:
    1265:    f3 0f 1e fb          endbr32
    1269:    55                  push    %ebp
    126a:    89 e5                mov     %esp,%ebp
    126c:    90                  nop
    126d:    90                  nop

```

将输出函数 nJolafiK 与 ppt 给出的程序框架对比，基本确定第一个 call 指令调用 strcmp 函数，第二个 call 指令调用 printf 函数，由于重定位尚未进行，这两处并未显示出具体的地址，一会可以用 gdb 查看。

先用 gdb 设断点查看输出函数 nJolafiK 的地址为 0x56556231:

```
Breakpoint 1, 0x56556231 in nJolafiK ()
```

写汇编代码使 do\_phase 调用输出函数 nJolafiK:

```

zr@ubuntu:~/shared/linklab-1190201421$ objdump -d call.o
call.o:          文件格式 elf32-i386

Disassembly of section .text:

00000000 <.text>:
   0:  b9 31 62 55 56          mov     $0x56556231,%ecx
   5:  ff d1                  call    *%ecx
zr@ubuntu:~/shared/linklab-1190201421$

```

1	movl	\$0x56556231,	%ecx
2	call	*%ecx	



```
1 movl 0x4(%esp), %ecx
2 lea 0xe6a(%ecx), %ecx
3 push %ecx
4 lea -0xe4b(%ecx), %ecx
5 call *%ecx
6 pop %ecx
```

[illegible]

```
zr@ubuntu:~/shared/linklab-1190201421$ gcc -m32 -o linkbomb2 main.o phase2.o
zr@ubuntu:~/shared/linklab-1190201421$ ./linkbomb2
1190201421
```

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

在实验中实际操作了对多个.o 文件的链接，对链接有了更深的理解，学会了用 readelf 查看 elf 相关信息，还在 phase2 中对重定位有了更深了解。

### 4.2 请给出对本次实验内容的建议

临近后半学期，学习压力较大，建议实验不要排的太紧，能多给点时间探索探索。

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>.