同济大学计算机系

计算机网络课程实验报告

学　　号 **1552239**

姓　　名 岳昊玮

专　　业 计算机科学与技术

授课老师 沈坚

一、 Linux 下的动态编译

动态编译的可执行文件需要附带一个的动态链接库，在执行时，需要调用其对应动态链接库中的命令。所以其优点一方面是缩小了执行文件本身的体积，另一方面是加快了编译速度，节省了系统资源。缺点一是哪怕是很简单的程序，只用到了链接库中的一两条命令，也需要附带一个相对庞大的链接库；二是如果其他计算机上没有安装对应的运行库，则用动态编译的可执行文件就不能运行。

```
[root@Anokoro 1552239-000103]# gcc -o helloc helloc.c
[root@Anokoro 1552239-000103]#
[root@Anokoro 1552239-000103]# g++ -o hellocpp hellocpp.cpp

[root@Anokoro 1552239-000103]#
[root@Anokoro 1552239-000103]# ls -l
总用量 36
drwxr-xr-x. 2 root root     6 10月   6 20:34 01
drwxr-xr-x. 2 root root     6 10月   6 20:34 02
-rwxr-xr-x. 1 root root  8512 10月   6 21:22 helloc
-rw-r--r--. 1 root root    72 10月   6 21:21 helloc.c
-rwxr-xr-x. 1 root root  9136 10月   6 21:23 hellocpp
-rw-r--r--. 1 root root    95 10月   6 21:21 hellocpp.cpp
-rw-r--r--. 1 root root    67 10月   6 20:56 test.c
[root@Anokoro 1552239-000103]#
```

printf("hello，world");程序的 gcc 动态编译命令，可执行文件字节 8512，cout<<"hello，world";程序的 c++/g++动态编译命令，可执行文件字节 9136。可以看出，c++的动态编译的可执行文件比 c 略大，

运行结果：

```
[root@Anokoro 1552239-000103]# /home/1552239-000103/helloc
hello, world
[root@Anokoro 1552239-000103]# /home/1552239-000103/hellocpp
hello, world
```

。

```
[root@Anokoro home]# g++ mysql_demo.cpp -o mysql_demo -L/usr/lib64/mysql -lmysqlclient
[root@Anokoro home]# ./mysql_demo
select return 4 records
学号：200215121   姓名：李勇    性别：男  年龄：20  系部：CS
学号：200215122   姓名：刘晨    性别：女  年龄：19  系部：CS
学号：200215123   姓名：王敏    性别：女  年龄：18  系部：MA
学号：200215125   姓名：张立    性别：男  年龄：19  系部：IS
[root@Anokoro home]# ls -l
总用量 2024
drwxr-xr-x. 5 root root      52 10月   4 18:05 1552239-000102
drwxr-xr-x. 4 root root     150 10月   6 21:27 1552239-000103
drwxr-xr-x. 4 root root      42 10月   5 20:13 1552239-000104
-rw-r--r--. 1 root root    1581 10月   5 21:45 1552239-linux-so.tar.bz2
-rw-r--r--. 1 root root    1138 9月   19 23:47 demo.php
-rw-r--r--. 1 root root     496 9月   19 23:47 demo.sql
-rw-r--r--. 1 root root 1609992 9月   29 00:05 glibc-static-2.17-196.el7.x86_64.rpm
-rw-r--r--. 1 root root  417824 9月   29 00:05 libstdc++-static-4.8.5-16.el7.x86_64.rpm
-rw-r--r--. 1 root root    2712 10月   5 18:26 linux-makefile.tar.bz2
-rwxr-xr-x. 1 root root   14272 10月   6 21:38 mysql_demo
-rw-r--r--. 1 root root    2261 9月   26 22:45 mysql_demo.cpp
[root@Anokoro home]#
```

如上图，mysql_demo 的可执行文件的字节数为 14272。

要想查找某个可执行文件所依赖的动态链接库，可以使用 ldd 命令，如下所示，以刚刚编写的 hello_c 为例

```
[root@Anokoro 1552239-000104]# ldd /home/1552239-000103/hello_c
        linux-vdso.so.1 =>  (0x00007ffea7085000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f8e986fd000)
        /lib64/ld-linux-x86-64.so.2 (0x00005563187d8000)
```

二、 Linux 下的 gcc 、 g++/c++静态编译

静态编译就是编译器在编译可执行文件的时候，将可执行文件需要调用的对应动态链接库(.so)中的部分提取出来，链接到可执行文件中去，使可执行文件在运行的时候不依赖于动态链接库。所以其优缺点与动态编译的可执行文件正好互补。

安装静态编译所需 rpm 包，直接使用 rpm 安装，通过 FlashFXP 将下载到的 rpm 包上传至虚拟机，使用 cd 命令进入 rpm 包所在的目录。如下操作即可

```
[root@Anokoro home]# rpm -ivh  glibc-static-2.17-196.el7.x86_64.rpm
警告：glibc-static-2.17-196.el7.x86_64.rpm: 头V3 RSA/SHA256 Signature, 密钥 ID f4a80eb5: NOKEY
准备中...                          ################################# [100%]
正在升级/安装...
   1:glibc-static-2.17-196.el7     ################################# [100%]
[root@Anokoro home]# rpm -ivh libstdc++-static-4.8.5-16.el7.x86_64.rpm
警告：libstdc++-static-4.8.5-16.el7.x86_64.rpm: 头V3 RSA/SHA256 Signature, 密钥 ID f4a80eb5: NOKEY
准备中...                          ################################# [100%]
正在升级/安装...
   1:libstdc++-static-4.8.5-16.el7 ################################# [100%]
```

```
[root@Anokoro 1552239-000103]# g++ hellocpp.cpp -static -o hellocpp_static
[root@Anokoro 1552239-000103]# gcc helloc.c -static -o helloc_static
[root@Anokoro 1552239-000103]#
[root@Anokoro 1552239-000103]# ls -l                    静态编译命令
总用量 2420
drwxr-xr-x. 2 root root       6 10月   6 20:34 01
drwxr-xr-x. 2 root root       6 10月   6 20:34 02
-rwxr-xr-x. 1 root root    8512 10月   6 21:22 helloc
-rw-r--r--. 1 root root      72 10月   6 21:21 helloc.c
-rwxr-xr-x. 1 root root    9136 10月   6 21:23 hellocpp
-rw-r--r--. 1 root root      95 10月   6 21:21 hellocpp.cpp
-rwxr-xr-x. 1 root root 1592208 10月   6 21:27 hellocpp_static
-rwxr-xr-x. 1 root root  844232 10月   6 21:27 helloc_static
-rw-r--r--. 1 root root      67 10月   6 20:56 test.c
```

和动态编译的唯一不同就是加上-static，从而表示静态编译。

printf("hello，world");程序的 gcc 静态编译命令，可执行文件字节 844232

cout << "hello,world";程序的 c++/g++静态编译命令，可执行文件字节 1592208

```
[root@Anokoro 1552239-000103]#
[root@Anokoro 1552239-000103]# ldd /home/1552239-000103/helloc_static
        不是动态可执行文件←当查找它所依赖的动态链接库时，提示不是动态可执行文件
[root@Anokoro 1552239-000103]# /home/1552239-000103/hellocpp_static
hello, world
[root@Anokoro 1552239-000103]# /home/1552239-000103/helloc_static
hello, world
[root@Anokoro 1552239-000103]#
```

结合上面的动态编译结果，无论是 c 还是 c++，静态编译的可执行文件都远远大于动态编译的可执行文件。而且，c 和 c++的静态编译可执行文件的大小也有了非常明显的差异。通过比较动态编译的可执行文件所依赖的动态链接库，可以发现，c++有 5 个，而 c 只要 3 个。多的两个分别是 libstdc++.so 和 libgcc_s.so

```
[root@Anokoro 1552239-000103]# ldd /home/1552239-000103/helloc
        linux-vdso.so.1 =>  (0x00007ffc2ddc1000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f2fb707f000)
        /lib64/ld-linux-x86-64.so.2 (0x0000563b333c2000)
[root@Anokoro 1552239-000103]# ldd /home/1552239-000103/hellocpp
        linux-vdso.so.1 =>  (0x00007ffdf9bfe000)
        libstdc++.so.6 => /lib64/libstdc++.so.6 (0x00007f605265a000)
        libm.so.6 => /lib64/libm.so.6 (0x00007f6052358000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007f6052141000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f6051d7e000)
        /lib64/ld-linux-x86-64.so.2 (0x00005604d1002000)
```

4.

4.2

makefile 内容

```
#使用的编译器
compiler = gcc
#编译的target目标
prom = test
#编译的源文件
src = $(shell find ./ -name "*.c")
obj = $(src:%.c=%.o)
$(prom): $(obj)
	$(compiler) -o $(prom) $(obj) -Wall -g -static
clean:
	-rm -rf $(obj) $(prom)
```

```
[root@Anokoro 01]# make
cc     -c -o test.o test.c
gcc -o test ./test.o  -wall -g -static
[root@Anokoro 01]# ls -l
总用量 840
-rw-r--r--. 1 root root    341 10月   6 13:52 makefile
-rwxr-xr-x. 1 root root 844224 10月   6 21:52 test
-rw-r--r--. 1 root root    103 10月   6 21:50 test.c
-rw-r--r--. 1 root root   1504 10月   6 21:52 test.o
[root@Anokoro 01]#
[root@Anokoro 01]#  /home/1552239-000103/01/test
1552239+岳昊玮
[root@Anokoro 01]#
[root@Anokoro 01]# make clean
rm -rf ./test.o  test
[root@Anokoro 01]#
```

4.3

makefile 内容

```
#使用的编译器
compiler = g++
#编译的target目标
prom = test
#编译的源文件
src = $(shell find ./ -name "*.cpp")
obj = $(src:%.cpp=%.o)
$(prom): $(obj)
	$(compiler) -o $(prom) $(obj) -Wall -g -static
clean:
	-rm -rf $(obj) $(prom)
```

```
[root@Anokoro 02]# make
g++     -c -o test.o test.cpp
g++ -o test ./test.o  -Wall -g -static
[root@Anokoro 02]#
[root@Anokoro 02]# ls -l
总用量 1568
-rw-r--r--. 1 root root     345 10月   6 13:57 makefile
-rwxr-xr-x. 1 root root 1592200 10月   6 21:57 test
-rw-r--r--. 1 root root     125 10月   6 21:56 test.cpp
-rw-r--r--. 1 root root    2480 10月   6 21:57 test.o
[root@Anokoro 02]#
[root@Anokoro 02]#  /home/1552239-000103/02/test
1552239+岳昊玮
[root@Anokoro 02]#
[root@Anokoro 02]# make clean
rm -rf ./test.o  test
[root@Anokoro 02]#
```

4.4

makefile 内容

```
# 需要排除的目录
exclude_dirs := include bin
# 取得当前子目录深度为1的所有目录名称
dirs := $(shell find . -maxdepth 1 -type d)
echo:$(dirs)
	echo $(dirs)
# basename 命令用于去掉路径信息，返回纯粹的文件名，如果指定的文件有扩展名，则将扩展名也一并去掉。
dirs := $(basename $(patsubst ./%,%,$(dirs)))
# filter-out 反过滤函数 和"filter"函数实现的功能相反。过滤掉字串"TEXT"中所有符合模式
dirs := $(filter-out $(exclude_dirs),$(dirs))

SUBDIRS := $(dirs)
# addprefix 添加前缀_clean_ ，避免clean子目录操作同名，
clean_dirs := $(addprefix _clean_,$(SUBDIRS) )

.PHONY: subdirs $(SUBDIRS) clean

# 执行默认make target
$(SUBDIRS):
	$(MAKE) -C $@
subdirs: $(SUBDIRS)

# 执行clean
$(clean_dirs):
	$(MAKE) -C $(patsubst _clean_%,%,$@) clean
clean: $(clean_dirs)
```

总有一种很奇怪的感觉，一个总 makefile 通关 3 次作业….

```
[root@Anokoro 1552239-000103]# make
make -C 01
make[1]: 进入目录“/home/1552239-000103/01”
cc    -c -o test.o test.c
gcc -o test ./test.o  -Wall -g -static
make[1]: 离开目录“/home/1552239-000103/01”
make -C 02
make[1]: 进入目录“/home/1552239-000103/02”
g++    -c -o test.o test.cpp
g++ -o test ./test.o  -Wall -g -static
make[1]: 离开目录“/home/1552239-000103/02”
echo 01 02
01 02
[root@Anokoro 1552239-000103]#
[root@Anokoro 1552239-000103]#
[root@Anokoro 1552239-000103]#
[root@Anokoro 1552239-000103]# make clean
make -C 01 clean
make[1]: 进入目录“/home/1552239-000103/01”
rm -rf ./test.o  test
make[1]: 离开目录“/home/1552239-000103/01”
make -C 02 clean
make[1]: 进入目录“/home/1552239-000103/02”
rm -rf ./test.o  test
make[1]: 离开目录“/home/1552239-000103/02”
[root@Anokoro 1552239-000103]#
```