

程序的静态与动态编译

1.Linux 下的动态编译

- 什么叫动态编译?

动态编译是某些程式语言在执行时用来增进效能的方法。尽管这技术源于 Self 但使用此技术最为人所知的是 Java。此技术可以做到一些只在执行时才能完成的最佳化。使用动态编译的执行环境一开始执行速度较慢，之后，完成大部分的编译和再编译后，会执行得比非动态编译程式快很多。因为初始化时的效能延迟，动态编译不适用于一些情况。在许多实作中，一些可以在编译时期做的最佳化被延到执行时期才编译，导致不必要的效能降低。即时编译是一种动态编译的形式。

- 给出printf("hello, world");程序的gcc动态编译命令，可执行文件字节数是多少?

```
[root@RHEL-zby test]# gcc -o hello hello.c
[root@RHEL-zby test]# ll
总用量 28
-rwxr-xr-x. 1 root root 8520 10月  5 17:31 hello
-rw-r--r--. 1 root root  68 9月  28 20:51 hello.c
-rw-r--r--. 1 root root 1552 10月  5 17:28 hello.o
-rwxr-xr-x. 1 root root 8008 10月  5 17:30 libhello.so
[root@RHEL-zby test]# ./hello
Hello world![root@RHEL-zby test]#
```

- 给出 `cout << "hello, world";`程序的 `c++/g++`动态编译命令，可执行文件字节数是多少？

```
[root@RHEL-zby test]# g++ -o hello-p hello.cpp
[root@RHEL-zby test]# ll
总用量 36
-rwxr-xr-x. 1 root root 8968 10月  5 17:33 hello-p
-rw-r--r--. 1 root root  68 9月  28 20:51 hello.c
-rw-r--r--. 1 root root  91 10月  5 17:33 hello.cpp
-rw-r--r--. 1 root root 1552 10月  5 17:28 hello.o
[root@RHEL-zby test]# ./hello-p
hello world!
```

- 给出第一周作业中mysql_demo.cpp的动态编译命令，可执行文件字节数是多少？

```
[root@RHEL-zby home]# g++ mysql_demo.cpp -o mysql -I/usr/include/mysql -L/usr/lib64/mysql -lmysqlclient
[root@RHEL-zby home]# ll
总用量 3968608
drwxr-xr-x. 5 root root      52 10月  2 23:29 1551265-000102
drwxr-xr-x. 4 root root      59 10月  5 16:04 1551265-000103
drwxr-xr-x. 4 root root      42 10月  2 23:29 1551265-000104
-rw-r--r--. 1 root root      613 9月  28 20:40 1551265-linux-static_compile.tar.bz2
-rwxr-xr-x. 1 root root    14272 10月  5 19:33 a.out
-rw-r--r--. 1 root root     1138 9月  28 18:56 demo.php
-rw-r--r--. 1 root root      496 9月  28 18:57 demo.sql
-rw-r--r--. 1 root root   1609992 10月  5 17:06 glibc-static-2.17-196.el7.x86_64.rpm
-rw-r--r--. 1 root root   2688240 10月  5 19:15 libmysql.so
-rw-r--r--. 1 root root   417824 10月  5 17:06 libstdc++-static-4.8.5-16.el7.x86_64.rpm
-rwxr-xr-x. 1 root root     14272 10月  5 19:37 mysql
-rw-r--r--. 1 root root      2259 9月  28 18:57 mysql_demo.cpp
-rw-r--r--. 1 root root     38500 10月  5 19:25 mysql.h
-rwxr-xr-x. 1 root root 4059037696 9月  28 18:45 rhel-server-7.4-x86_64-dvd.iso
drwxr-xr-x. 2 root root      81 10月  5 17:34 test
[root@RHEL-zby home]# ./mysql
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
```

● 如何查找某个可执行文件所依赖的动态链接库?

ldd 命令可以查看一个可执行程序依赖的共享库,
例如# ldd /bin/lnlibc.so.6
=> /lib/libc.so.6 (0x40021000)/lib/ld-linux.so.2
=> /lib/ld- linux.so.2 (0x40000000)
可以看到 ln 命令依赖于 libc 库和 ld-linux 库

2.Linux 下的 gcc 静态编译

● 什么叫静态编译?

通常情况下, 对函数库的链接是放在编译时期 (**compile time**) 完成的。所有相关的对象文件 (**object file**) 与牵涉到的函数库 (**library**) 被链接合成一个可执行文件 (**executable file**)。程序在运行时, 与函数库再无瓜葛, 因为所有需要的函数已拷贝到自己门下。所以这些函数库被成为静态库 (**static libaray**), 通常文件名为“libxxx.a”的形式。

● 给出 printf("hello, world");程序的 gcc 静态编译命令, 可执行文件字节数是多少?

先装个包

```
[root@RHEL-zby home]# rpm -i 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
```

静态编译

```
[root@RHEL-zby test]# gcc hello.c -static -o hello  
[root@RHEL-zby test]# ll  
总用量 832  
-rwxr-xr-x. 1 root root 844152 10月  5 17:07 hello  
-rw-r--r--. 1 root root    68 9月  28 20:51 hello.c
```


3. Linux 下的 c++/g++静态编译

装一个包

```
[root@RHEL-zby home]# rpm -i 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 f4a8
0eb5: NOKEY
```

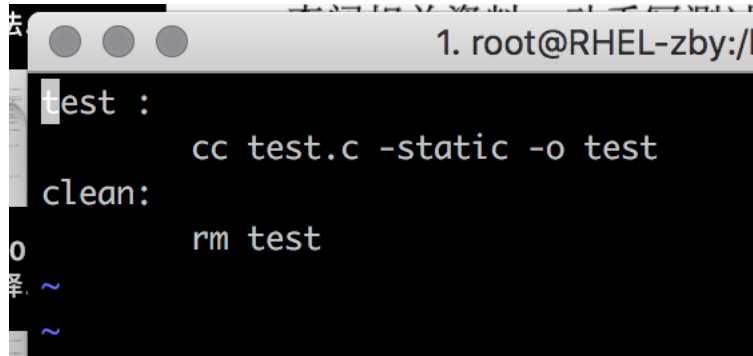
静态编译

```
[root@RHEL-zby test]# g++ hello.cpp -static -o hello-s
[root@RHEL-zby test]# ll
总用量 1580
-rwxr-xr-x. 1 root root 8968 10月 5 17:33 hello
-rw-r--r--. 1 root root 68 9月 28 20:51 hello.c
-rw-r--r--. 1 root root 91 10月 5 17:33 hello.cpp
-rw-r--r--. 1 root root 1552 10月 5 17:28 hello.o
-rwxr-xr-x. 1 root root 1592200 10月 7 20:04 hello-s
[root@RHEL-zby test]# ./hello-s
hello world![root@RHEL-zby test]#
```



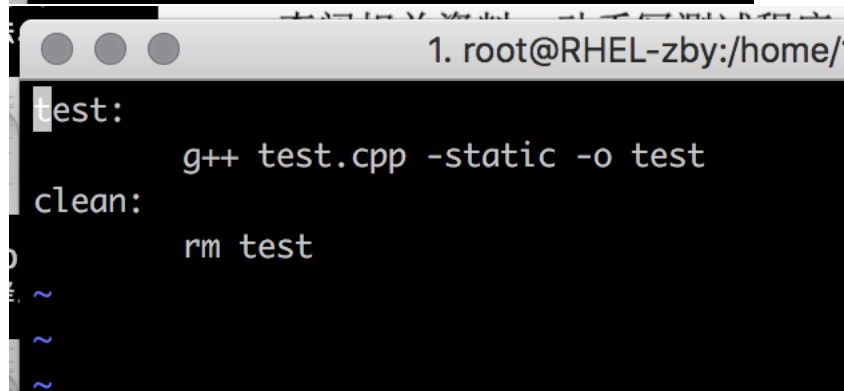
4. 按要求写出下列几种常用情况的静态编译测试样例

同 makefile 的文档 细节不再多写
只是用 g++ 编译的地方稍微改一下



A terminal window with a title bar showing '1. root@RHEL-zby:/l'. The terminal displays a Makefile rule for 'test' using 'cc' to compile 'test.c' into a static binary 'test'. It also includes a 'clean' rule to remove 'test'.

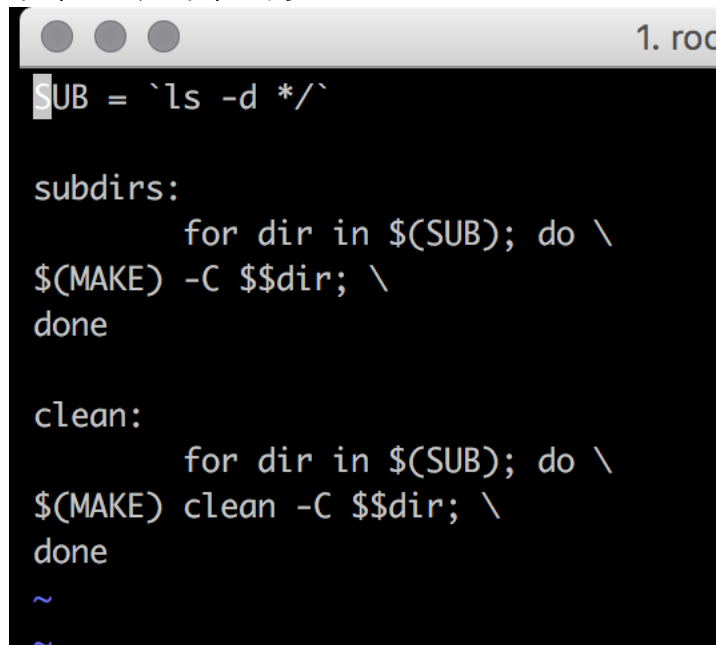
```
test :  
    cc test.c -static -o test  
clean:  
    rm test  
~  
~
```



A terminal window with a title bar showing '1. root@RHEL-zby:/home/'. The terminal displays a Makefile rule for 'test' using 'g++' to compile 'test.cpp' into a static binary 'test'. It also includes a 'clean' rule to remove 'test'.

```
test:  
    g++ test.cpp -static -o test  
clean:  
    rm test  
~  
~  
~
```

最外层还是那个总的 makefile



A terminal window with a title bar showing '1. roc'. The terminal displays a Makefile rule for 'subdirs' that iterates over all subdirectories and runs 'make' in each. It also includes a 'clean' rule that iterates over all subdirectories and runs 'make clean' in each.

```
SUB = `ls -d */`  
  
subdirs:  
    for dir in $(SUB); do \  
$(MAKE) -C $$dir; \  
done  
  
clean:  
    for dir in $(SUB); do \  
$(MAKE) clean -C $$dir; \  
done  
~  
~
```