<http://blog.chinaunix.net/uid-261392-id-2138941.html>

我们通常把一些公用函数制作成函数库，供其它程序使用。函数库分为静态库和动态库两  
种。静态库在程序编译时会被连接到目标代码中，程序运行时将不再需要该静态库。动态  
库在程序编译时并不会被连接到目标代码中，而是在程序运行是才被载入，因此在程序运  
行时还需要动态库存在。本文主要通过举例来说明在Linux中如何创建静态库和动态库，以  
及使用它们。

在创建函数库前，我们先来准备举例用的源程序，并将函数库的源程序编译成.o文件。

第1步：编辑得到举例的程序--hello.h、hello.c和main.c；

hello.c(见程序2)是函数库的源程序，其中包含公用函数hello，该函数将在屏幕上输出"  
Hello XXX!"。hello.h(见程序1)为该函数库的头文件。main.c(见程序3)为测试库文件的  
主程序，在主程序中调用了公用函数hello。

#ifndef HELLO\_H  
#define HELLO\_H

void hello(const char \*name);

#endif //HELLO\_H  
程序1: hello.h

#include

void hello(const char \*name)  
{  
printf("Hello %s!\n", name);  
}  
程序2: hello.c

#include "hello.h"

int main()  
{  
hello("everyone");  
return 0;  
}  
程序3: main.c

第2步：将hello.c编译成.o文件；

无论静态库，还是动态库，都是由.o文件创建的。因此，我们必须将源程序hello.c通过g  
cc先编译成.o文件。

在系统提示符下键入以下命令得到hello.o文件。

# gcc -c hello.c

#

我们运行ls命令看看是否生存了hello.o文件。

# ls

hello.c hello.h hello.o main.c

#

在ls命令结果中，我们看到了hello.o文件，本步操作完成。

下面我们先来看看如何创建静态库，以及使用它。

第3步：由.o文件创建静态库；

静态库文件名的命名规范是以lib为前缀，紧接着跟静态库名，扩展名为.a。例如：我们将  
创建的静态库名为myhello，则静态库文件名就是libmyhello.a。在创建和使用静态库时，  
需要注意这点。创建静态库用ar命令。

在系统提示符下键入以下命令将创建静态库文件libmyhello.a。

# ar crv libmyhello.a hello.o

#

我们同样运行ls命令查看结果：

# ls

hello.c hello.h hello.o libmyhello.a main.c

#

ls命令结果中有libmyhello.a。

第4步：在程序中使用静态库；

静态库制作完了，如何使用它内部的函数呢？只需要在使用到这些公用函数的源程序中包  
含这些公用函数的原型声明，然后在用gcc命令生成目标文件时指明静态库名，gcc将会从  
静态库中将公用函数连接到目标文件中。注意，gcc会在静态库名前加上前缀lib，然后追  
加扩展名.a得到的静态库文件名来查找静态库文件。

在程序3:main.c中，我们包含了静态库的头文件hello.h，然后在主程序main中直接调用公  
用函数hello。下面先生成目标程序hello，然后运行hello程序看看结果如何。

(# gcc -o hello main.c -L. -lmyhello??)

#gcc main.c libmyhello.a -o main

# ./hello

Hello everyone!

#

我们删除静态库文件试试公用函数hello是否真的连接到目标文件 hello中了。

# rm libmyhello.a

rm: remove regular file `libmyhello.a'? y

# ./hello

Hello everyone!

#

程序照常运行，静态库中的公用函数已经连接到目标文件中了。

我们继续看看如何在Linux中创建动态库。我们还是从.o文件开始。

第5步：由.o文件创建动态库文件；

动态库文件名命名规范和静态库文件名命名规范类似，也是在动态库名增加前缀lib，但其  
文件扩展名为.so。例如：我们将创建的动态库名为myhello，则动态库文件名就是libmyh  
ello.so。用gcc来创建动态库。

在系统提示符下键入以下命令得到动态库文件libmyhello.so。

# gcc -shared -fPCI -o libmyhello.so hello.o

#

我们照样使用ls命令看看动态库文件是否生成。

# ls

hello.c hello.h hello.o libmyhello.so main.c

#

第6步：在程序中使用动态库；

在程序中使用动态库和使用静态库完全一样，也是在使用到这些公用函数的源程序中包含  
这些公用函数的原型声明，然后在用gcc命令生成目标文件时指明动态库名进行编译。我们  
先运行gcc命令生成目标文件，再运行它看看结果。

# gcc -o hello main.c -L. -lmyhello

# ./hello

./hello: error while loading shared libraries: libmyhello.so: cannot open shar  
ed object file: No such file or directory

#

哦！出错了。快看看错误提示，原来是找不到动态库文件libmyhello.so。程序在运行时，  
会在/usr/lib和/lib等目录中查找需要的动态库文件。若找到，则载入动态库，否则将提  
示类似上述错误而终止程序运行。我们将文件libmyhello.so复制到目录/usr/lib中，再试  
试。

# mv libmyhello.so /usr/lib

# ./hello

Hello everyone!

#

成功了。这也进一步说明了动态库在程序运行时是需要的。

我们回过头看看，发现使用静态库和使用动态库编译成目标程序使用的gcc命令完全一样，  
那当静态库和动态库同名时，gcc命令会使用哪个库文件呢？抱着对问题必究到底的心情，  
来试试看。

先删除除.c和.h外的所有文件，恢复成我们刚刚编辑完举例程序状态。

# rm -f hello hello.o /usr/lib/libmyhello.so

# ls

hello.c hello.h main.c

#

在来创建静态库文件libmyhello.a和动态库文件libmyhello.so。

# gcc -c hello.c

# ar cr libmyhello.a hello.o

# gcc -shared -fPCI -o libmyhello.so hello.o

# ls

hello.c hello.h hello.o libmyhello.a libmyhello.so main.c

#

通过上述最后一条ls命令，可以发现静态库文件libmyhello.a和动态库文件libmyhello.s  
o都已经生成，并都在当前目录中。然后，我们运行gcc命令来使用函数库myhello生成目标  
文件hello，并运行程序 hello。

# gcc -o hello main.c -L. -lmyhello

# ./hello

./hello: error while loading shared libraries: libmyhello.so: cannot open shar  
ed object file: No such file or directory

#

从程序hello运行的结果中很容易知道，当静态库和动态库同名时， gcc命令将优先使用动  
态库。

Note:  
编译参数解析  
最主要的是GCC命令行的一个选项:  
-shared 该选项指定生成动态连接库（让连接器生成T类型的导出符号表，有时候也生成弱连接W类型的导出符号），不用该标志外部程序无法连接。相当于一个可执行文件  
l -fPIC：表示编译为位置独立的代码，不用此选项的话编译后的代码是位置相关的所以动态载入时是通过代码拷贝的方式来满足不同进程的需要，而不能达到真正代码段共享的目的。  
l -L.：表示要连接的库在当前目录中  
l -ltest：编译器查找动态连接库时有隐含的命名规则，即在给出的名字前面加上lib，后面加上.so来确定库的名称  
l LD\_LIBRARY\_PATH：这个环境变量指示动态连接器可以装载动态库的路径。  
l 当然如果有root权限的话，可以修改/etc/ld.so.conf文件，然后调用 /sbin/ldconfig来达到同样的目的，不过如果没有root权限，那么只能采用输出LD\_LIBRARY\_PATH的方法了。

调用动态库的时候有几个问题会经常碰到，有时，明明已经将库的头文件所在目录 通过 “-I” include进来了，库所在文件通过 “-L”参数引导，并指定了“-l”的库名，但通过ldd命令察看时，就是死活找不到你指定链接的so文件，这时你要作的就是通过修改 LD\_LIBRARY\_PATH或者/etc/ld.so.conf文件来指定动态库的目录。通常这样做就可以解决库无法链接的问题了。