<https://www.cnblogs.com/qingjiaowoxiaoxioashou/p/6410588.html>

# [如何使用GCC生成动态库和静态库](http://www.cnblogs.com/qingjiaowoxiaoxioashou/p/6410588.html)

　　根据链接时期的不同，库又有静态库和动态库之分。静态库是在链接阶段被链接的，所以生成的可执行文件就不受库的影响，即使库被删除，程序依然可以成功运行。而动态库是在程序执行的时候被链接的。程序执行完，库仍需保留在系统上，以供程序运行时调用。链接静态库从某种意义上来说是一种复制粘贴，被链接后库就直接嵌入可执行程序中了，这样系统空间有很大的浪费，而且一旦发现系统中有bug，就必须一一把链接该库的程序找出来，然后重新编译，十分麻烦。而动态库刚好弥补了这个缺陷，因为动态库是在程序运行时被链接的，所以磁盘上只需保留一份副本，一次节约了空间，如果发现bug或者是要升级，只要用新的库把原来的替换掉就可以了。静态库是不是一无是处了呢？非也。如果代码在其他系统上运行，且没有相应的库时，解决办法就是使用静态库。而且由于动态库是在程序运行的时候被链接，因此动态库的运行速度比较慢。

　　好了，我们了解了关于动态库和静态库的相关知识，那么如何使用GCC生成静态库和动态库呢？

　　我们参考了《LinuxC程序设计大全》上面的例子，来总结GCC下编译静态及其动态链接库的方法及步骤。

　　程序清单如下：

**test.c**

　　int add(int a,int b)

     {

　　    retrun a+b;

     }

     int sub(int a,int b)

     {

　　    retrun a-b;

     }

     int mul(int a,int b)

     {

　　    retrun a\*b;

     }

     int div(int a,int b)

     {

　　    retrun a/b;

     }

**test.h的内容**

　　#ifndef \_TEST\_H\_

　　#define \_TEST\_H\_

　　extern int add(int a,int b);

　　extern int sub(int a,int b);

　　extern int mul(int a,int b);

　　extern int div(int a,int b);

**main.c文件内容**

　　#include<stduo.h>

　　#include<test.h>

　　int main()

　　{

　　　　int a,b;

　　　　printf("please input a and b\n");

　　　　scanf("%d %d",&a,&b);

　　　　printf("The add:%d\n",add(a,b));

　　　　printf("The sub:%d\n",sub(a,b));

　　　　printf("The mul:%d\n",mul(a,b));

　　　　printf("The div:%d\n",div(a,b));

　　}

**1.使用gcc生成静态库及静态库使用方法**：

　　在此例中，test.c用于编译生成静态库libtest.a，test.h为libtest.a对应的头文件。

　　第一步：生成test.o目标文件，使用gcc -c test.c -o test.o命令。

　　第二步：使用ar将test.o打包成libtest.a静态库，使用ar rcs -o libtest.a test.o命令

　　第三步：生成libtest.a静态库后，可以使用命令ar t libtest.a查看libtest.a文件中包含哪些文件。

　　第四步：编译main.c，并使用libtest.a静态库，链接时-l参数后不加空格指定所需要链接的库，这里库名是libtest.a，但是只需要给出-ltest即可，ld会以libtest作为库的实际名字。完整的命令为：gcc -o app\_static main.c -L. -ltest 或者是gcc -o app\_static main.c libtest.a

　　第五步：运行app\_static

　　直接使用命令./app\_static

**2.使用gcc生成动态库及使用动态库的方法**

　　第一步：生成test.o目标文件，使用如下命令。在此处需要添加-fPIC参数，该参数用于生成位置无关代码已工生成动态库使用，使用命令：gcc -c -o test.o -fPIC test.c

　　第二步：使用-shared参数生成动态库，使用如下命令：gcc -shared -o libmyshare.so test.o,上述两个命令可以连在一起，如下所示：gcc -shared -fPIC -o libmyshare.so test.c

　　第三步：编译main.c，使用libmyshare.so动态库，命令如下gcc -o app\_share main.c -L. -lmyshare.使用ldd app\_share命令查看app\_share使用动态库，如果libmyshare无法找到，直接执行app\_share就会出现错误。解决方法：首先使用export LD\_LIBRARY\_PATH=.:$LD\_LIBRARY\_PATH将当前目录加入LD\_LIBRARY\_PATH变量中。再次运行ldd app\_share

　　另一种编译main.c，并链接libmyshare.so的方式如下（该方式通过./libmyshare.so直接指定使用当前目录下的libmyshare.so文件），使用命令：gcc -o app\_share main.c ./libmyshare.so