<https://blog.csdn.net/a600423444/article/details/7206015>

在windows下动态链接库是以.dll后缀的文件，二在Linux中，是以.so作后缀的文件。  
动态链接库的好处就是节省内存空间。  
  
  
1、Linux下创建动态链接库  
在使用GCC编译程序时，只需加上-shared选项即可，这样生成的执行程序即为动态链接库。  
例如有文件：hello.c x.h main.c

**[plain]** [view plain](https://blog.csdn.net/a600423444/article/details/7206015) [copy](https://blog.csdn.net/a600423444/article/details/7206015)

1. 编译：gcc hello.c -fPIC -o libhello.so

其中-fPIC选项的作用是：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的调用，而不能达到真正的代码段共享的目的.  
  
  
将main.c与hello.so动态库

**[plain]** [view plain](https://blog.csdn.net/a600423444/article/details/7206015) [copy](https://blog.csdn.net/a600423444/article/details/7206015)

1. gcc main.c -L. -lhello -o main

一、动态链接库

1.创建hello.so动态库

**[cpp]** [view plain](https://blog.csdn.net/a600423444/article/details/7206015) [copy](https://blog.csdn.net/a600423444/article/details/7206015)

1. #include <stdio.h>
2. **void** hello(){
3. printf("hello world\n");
4. }
5. 编译：gcc -fPIC -shared hello.c -o libhello.so

2.hello.h头文件

**[cpp]** [view plain](https://blog.csdn.net/a600423444/article/details/7206015) [copy](https://blog.csdn.net/a600423444/article/details/7206015)

1. **void** hello();

3.链接动态库

**[cpp]** [view plain](https://blog.csdn.net/a600423444/article/details/7206015) [copy](https://blog.csdn.net/a600423444/article/details/7206015)

1. #include <stdio.h>
2. #include "hello.h"
4. **int** main(){
5. printf("call hello()");
6. hello();
7. }
8. 编译：gcc main.c -L. -lhello -o main

这里-L的选项是指定编译器在搜索动态库时搜索的路径，告诉编译器hello库的位置。"."意思是当前路径.

3.编译成够后执行./main，会提示：

**[plain]** [view plain](https://blog.csdn.net/a600423444/article/details/7206015) [copy](https://blog.csdn.net/a600423444/article/details/7206015)

1. In function `main':
3. main.c:(.text+0x1d): undefined reference to `hello'
4. collect2: ld returned 1 exit status

这是因为在链接hello动态库时，编译器没有找到。  
解决方法：

**[plain]** [view plain](https://blog.csdn.net/a600423444/article/details/7206015) [copy](https://blog.csdn.net/a600423444/article/details/7206015)

1. sudo cp libhello.so /usr/lib/

这样，再次执行就成功输入:  
call hello()

二、静态库

文件有：main.c、hello.c、hello.h  
1.编译静态库hello.o:

**[plain]** [view plain](https://blog.csdn.net/a600423444/article/details/7206015) [copy](https://blog.csdn.net/a600423444/article/details/7206015)

1. gcc hello.c -o hello.o  #这里没有使用-shared

2.把目标文档归档

**[plain]** [view plain](https://blog.csdn.net/a600423444/article/details/7206015) [copy](https://blog.csdn.net/a600423444/article/details/7206015)

1. ar -r libhello.a hello.o  #这里的ar相当于tar的作用，将多个目标打包。

程序ar配合参数-r创建一个新库libhello.a，并将命令行中列出的文件打包入其中。这种方法，如果libhello.a已经存在，将会覆盖现在文件，否则将新创建。  
  
3.链接静态库

**[plain]** [view plain](https://blog.csdn.net/a600423444/article/details/7206015) [copy](https://blog.csdn.net/a600423444/article/details/7206015)

1. gcc main.c -lhello -L. -static -o main

这里的-static选项是告诉编译器,hello是静态库。  
或者：

**[plain]** [view plain](https://blog.csdn.net/a600423444/article/details/7206015) [copy](https://blog.csdn.net/a600423444/article/details/7206015)

1. gcc main.c libhello.a -L. -o main

这样就可以不用加-static

4.执行./main

输出：call hello()

三、借助自带的ldd实现程序来分析动态库搜索情况

ldd main

结果：  
linux-gate.so.1 =>  (0x00efd000)  
libhello.so => /usr/lib/libhello.so (0x00f6b000)  
libc.so.6 => /lib/libc.so.6 (0x001a5000)  
/lib/ld-linux.so.2 (0x00eb8000)  
如果目标程序没有链接动态库，则打印“not a dynamic executable”