<https://blog.csdn.net/mybelief321/article/details/9099659>

本文详细介绍了linux 下gcc头文件指定方法，以及搜索路径顺序的问题。另外，还总结了，gcc动态链接的方法以及路径指定，同样也讨论了搜索路径的顺序问题。本文包含了很多的例子，具有很强的操作性，希望读者自己去走一遍。  
一.#include <>与#include “”

#include <>直接到系统指定的某些目录中去找某些头文件。  
#include “”先到源文件所在文件夹去找，然后再到系统指定的某些目录中去找某些头文件。

二.gcc指定头文件的三种情况：

1.会在默认情况下指定到/usr/include文件夹(更深层次的是一个相对路径，gcc可执行程序的路径是/usr/bin/gcc，那么它在实际工作时指定头文件头径是一种相对路径方法，换算成绝对路径就是加上/usr/include，如#include 就是包含/usr/include/stdio.h)

2.GCC还使用了-I指定路径的方式，即  
gcc -I 头文件所在文件夹(绝对路径或相对路径均可) 源文件  
举一个例子：  
设当前路径为/root/test,其结构如下：  
include\_test.c  
include/include\_test.h  
有两种方法访问到include\_test.h。  
1. include\_test.c中#include “include/include\_test.h”然后gcc include\_test.c即可  
2. include\_test.c中#include 或者#include 然后gcc –I include include\_test.c也可

3. 参数：-nostdinc使编译器不再系统缺省的头文件目录里面找头文件,一般和-I联合使用,明确限定头文件的位置。

在编译驱动模块时，由于非凡的需求必须强制GCC不搜索系统默认路径，也就是不搜索/usr/include要用参数-nostdinc，还要自己用-I参数来指定内核头文件路径，这个时候必须在Makefile中指定。

头文件搜索顺序：  
1.由参数-I指定的路径(指定路径有多个路径时，按指定路径的顺序搜索)

2.然后找gcc的环境变量 C\_INCLUDE\_PATH, CPLUS\_INCLUDE\_PATH, OBJC\_INCLUDE\_PATH

3.再找内定目录  
/usr/include  
/usr/local/include  
/usr/lib/gcc-lib/i386-linux/2.95.2/include  
/usr/lib/gcc-lib/i386-linux/2.95.2/../../../../include/g++-3  
/usr/lib/gcc-lib/i386-linux/2.95.2/../../../../i386-linux/include

库文件，但是如果装gcc的时候，是有给定的prefix的话，那么就是  
/usr/include  
prefix/include  
prefix/xxx-xxx-xxx-gnulibc/include  
prefix/lib/gcc-lib/xxxx-xxx-xxx-gnulibc/2.8.1/include

三.Linux指定动态库路径

众所周知，Linux动态库的默认搜索路径是/lib和/usr/lib。动态库被创建后，一般都复制到这两个目录中。当程序执行时需要某动态库， 并且该动态库还未加载到内存中，则系统会自动到这两个默认搜索路径中去查找相应的动态库文件，然后加载该文件到内存中，这样程序就可以使用该动态库中的函 数，以及该动态库的其它资源了。在Linux 中，动态库的搜索路径除了默认的搜索路径外，还可以通过以下三种方法来指定。

1.在配置文件/etc/ld.so.conf中指定动态库搜索路径。  
可以通过编辑配置文件/etc/ld.so.conf来指定动态库的搜索路径，该文件中每行为一个动态库搜索路径。每次编辑完该文件后，都必须运行命令ldconfig使修改后的配置生效。

举一个例子：  
所有源文件：  
源文件1: lib\_test.c  
#include   
void prt()  
{  
printf(“You found me!!!/n”);  
}  
源文件2: main.c  
void prt();  
int main()  
{  
prt();  
return 0;  
}  
操作过程：  
我们通过以下命令用源程序lib\_test.c来创建动态库 lib\_test.so。  
# gcc –o lib\_test.o -c lib\_test.c  
# gcc -shared -fPIC -o lib\_test.so lib\_test.o  
#  
或者直接一条指令：  
#gcc –shared –fPIC –o lib\_test.so lib\_test.c  
#

注意：  
-fPIC参数声明链接库的代码段是可以共享的，  
-shared参数声明编译为共享库。请注意这次我们编译的共享库的名字叫做  
lib\_test.so，这也是Linux共享库的一个命名的惯例了：后缀使用so，而名称使用libxxxx格式。

接着通过以下命令编译main.c,生成目标程序main.out。  
# gcc -o main.out -L. –l\_test main.c  
#

请注意为什么是-l\_test?

然后把库文件移动到目录/root/lib中。  
# mkdir /root/lib  
# mv lib\_test.so /root/lib/ lib\_test.so  
#

最后编辑配置文件/etc/ld.so.conf，在该文件中追加一行/root/lib。

运行程序main.out:  
# ./main.out  
./main.out: error while loading shared libraries: lib\_test.so: cannot open shared object file: No such file or directory  
#  
出错了，系统未找到动态库lib\_test.so。找找原因，原来在编辑完配置文件/etc/ld.so.conf后，没有运行命令ldconfig，所以刚才的修改还未生效。我们运行ldconfig后再试试。

# ldconfig  
# ./main.out  
You found me!!!  
#  
程序main.out运行成功，并且打印出正确结果。

2.通过环境变量LD\_LIBRARY\_PATH指定动态库搜索路径。  
通过设定环境变量LD\_LIBRARY\_PATH也可以指定动态库搜索路径。当通过该环境变量指定多个动态库搜索路径时，路径之间用冒号”:”分隔。下面通过例2来说明本方法。

举一个例子：  
这次我们把上面得到的文件lib\_test.so移动到另一个地方去，如/root下面，然后设置环境变量LD\_LIBRARY\_PATH找到lib\_test.so。设置环境变量方法如下：  
# export LD\_LIBRARY\_PATH=/root  
#  
然后运行：  
#./main.out  
You found me!!!  
#  
注意：设置环境变量LD\_LIBRARY\_PATH=/root是不行的，非得export才行。

3.在编译目标代码时指定该程序的动态库搜索路径。  
还可以在编译目标代码时指定程序的动态库搜索路径。-Wl,表示后面的参数将传给link程序ld（因为gcc可能会自动调用ld）。这里通过gcc 的参数”-Wl,-rpath,”指定  
举一个例子：  
这次我们还把上面得到的文件lib\_test.so移动到另一个地方去，如/root/test/lib下面，  
因为我们需要在编译目标代码时指定可执行文件的动态库搜索路径，所以需要用gcc命令重新编译源程序main.c(见程序2)来生成可执行文件main.out。  
# gcc -o main.out -L. –l\_test -Wl,-rpath,/root/test/lib main.c  
#

运行结果：  
# ./main.out  
You found me!!!  
#

程序./main.out运行成功，输出的结果正是main.c中的函数prt的运行结果。因此程序main.out搜索到的动态库是/root/test/lib/lib\_test.so。

关于-Wl,rpath的使用方法我再举一个例子，应该不难从中看出指定多个路径的方法：  
gcc -Wl,-rpath,/home/arc/test,-rpath,/lib/,-rpath,/usr/lib/,-rpath,/usr/local/lib test.c

以上介绍了三种指定动态库搜索路径的方法，加上默认的动态库搜索路径/lib和/usr/lib，共五种动态库的搜索路径，那么它们搜索的先后顺序是什么呢？读者可以用下面的方法来试验一下：  
（1） 用前面介绍的方法生成5个lib\_test.so放在5个不同的文件夹下面，要求每一个lib\_test.so都唯一对应一个搜索路径，并注意main.out程序输出的不同。  
（2） 运行main.out，即可看出他是那个搜索路径下的，然后删除这个路径下的lib\_test.so，然后再运行。依此类推操作，即可推出搜索顺序。

可以得出动态库的搜索路径搜索的先后顺序是：

1.编译目标代码时指定的动态库搜索路径；

2.环境变量LD\_LIBRARY\_PATH指定的动态库搜索路径；

3.配置文件/etc/ld.so.conf中指定的动态库搜索路径；

4.默认的动态库搜索路径/lib；

5.默认的动态库搜索路径/usr/lib。

在上述1、2、3指定动态库搜索路径时，都可指定多个动态库搜索路径，其搜索的先后顺序是按指定路径的先后顺序搜索的。有兴趣的读者自己验证。

PS：此文网上搜得，原始出处已经无法访问，故无法给出原文链接。

转自：<http://wdqfirst.blog.163.com/blog/static/1133474112012108114838299/>

个人分类： [Linux系统学习](https://blog.csdn.net/mybelief321/article/category/1431936)