<https://blog.csdn.net/casularm/article/details/316149>

作者： www.linuxfans.org mozilla

//链接多个目录下的动态库

-L/usr/lib -L/usr/X11R6/lib -lgtk -lgdk -rdynamic -lgmodule -lglib -ldl -lXi -lXext -lX11 -lm

1。gcc包含的c/c++编译器  
gcc,cc,c++,g++,gcc和cc是一样的，c++和g++是一样的，(没有看太明白前面这半句是什  
么意思:))一般c程序就用gcc编译，c++程序就用g++编译  
  
2。gcc的基本用法  
gcc test.c这样将编译出一个名为a.out的程序  
gcc test.c -o test这样将编译出一个名为test的程序，-o参数用来指定生成程序的名  
字  
  
3。为什么会出现undefined reference to 'xxxxx'错误？  
首先这是链接错误，不是编译错误，也就是说如果只有这个错误，说明你的程序源码本  
身没有问题，是你用编译器编译时参数用得不对，你没  
  
有指定链接程序要用到得库，比如你的程序里用到了一些数学函数，那么你就要在编译  
参数里指定程序要链接数学库，方法是在编译命令行里加入-lm。  
  
4。-l参数和-L参数  
-l参数就是用来指定程序要链接的库，-l参数紧接着就是库名，那么库名跟真正的库文  
件名有什么关系呢？  
就拿数学库来说，他的库名是m，他的库文件名是libm.so，很容易看出，把库文件名的  
头lib和尾.so去掉就是库名了。  
  
好了现在我们知道怎么得到库名了，比如我们自已要用到一个第三方提供的库名字叫lib  
test.so，那么我们只要把libtest.so拷贝到/usr/lib  
  
里，编译时加上-ltest参数，我们就能用上libtest.so库了（当然要用libtest.so库里  
的函数，我们还需要与libtest.so配套的头文件）。  
  
放在/lib和/usr/lib和/usr/local/lib里的库直接用-l参数就能链接了，但如果库文件  
没放在这三个目录里，而是放在其他目录里，这时我们  
  
只用-l参数的话，链接还是会出错，出错信息大概是：“/usr/bin/ld: cannot find   
-lxxx”，也就是链接程序ld在那3个目录里找不到  
  
libxxx.so，这时另外一个参数-L就派上用场了，比如常用的X11的库，它放在/usr/X11R  
6/lib目录下，我们编译时就要用-L/usr/X11R6/lib -  
  
lX11参数，-L参数跟着的是库文件所在的目录名。再比如我们把libtest.so放在/aaa/bb  
b/ccc目录下，那链接参数就是-L/aaa/bbb/ccc -ltest  
  
另外，大部分libxxxx.so只是一个链接，以RH9为例，比如libm.so它链接到/lib/libm.s  
o.x，/lib/libm.so.6又链接到/lib/libm-2.3.2.so，  
  
如果没有这样的链接，还是会出错，因为ld只会找libxxxx.so，所以如果你要用到xxxx  
库，而只有libxxxx.so.x或者libxxxx-x.x.x.so，做一  
  
个链接就可以了ln -s libxxxx-x.x.x.so libxxxx.so  
  
手工来写链接参数总是很麻烦的，还好很多库开发包提供了生成链接参数的程序，名字  
一般叫xxxx-config，一般放在/usr/bin目录下，比如  
  
gtk1.2的链接参数生成程序是gtk-config，执行gtk-config --libs就能得到以下输出"-  
L/usr/lib -L/usr/X11R6/lib -lgtk -lgdk -rdynamic   
  
-lgmodule -lglib -ldl -lXi -lXext -lX11 -lm"，这就是编译一个gtk1.2程序所需的g  
tk链接参数，xxx-config除了--libs参数外还有一个参  
  
数是--cflags用来生成头文  
件包含目录的，也就是-I参数，在下面我们将会讲到。你可以试试执行gtk-config   
--libs --cflags，看看输出结果。  
现在的问题就是怎样用这些输出结果了，最笨的方法就是复制粘贴或者照抄，聪明的办  
法是在编译命令行里加入这个`xxxx-config --libs --  
  
cflags`，比如编译一个gtk程序：gcc gtktest.c `gtk-config --libs --cflags`这样  
就差  
不多了。注意`不是单引号，而是1键左边那个键。  
  
除了xxx-config以外，现在新的开发包一般都用pkg-config来生成链接参数，使用方法  
跟xxx-config类似，但xxx-config是针对特定的开发包  
  
，但pkg-config包含很多开发包的链接参数的生成，用pkg-config --list-all命令可以  
列出所支持的所有开发包，pkg-config的用法就是pkg  
  
-config pagName --libs --cflags，其中pagName是包名，是pkg-config--list-all里  
列出名单中的一个，比如gtk1.2的名字就是gtk+，pkg-  
  
config gtk+ --libs --cflags的作用跟gtk-config --libs --cflags是一样的。比如：  
gcc gtktest.c `pkg-config gtk+ --libs --cflags`  
  
。  
  
5。-include和-I参数  
-include用来包含头文件，但一般情况下包含头文件都在源码里用#include xxxxxx实现  
，-include参数很少用。-I参数是用来指定头文件目录  
  
，/usr/include目录一般是不用指定的，gcc知道去那里找，但是如果头文件不在/usr/i  
nclude里我们就要用-I参数指定了，比如头文件放  
  
在/myinclude目录里，那编译命令行就要加上-I/myinclude参数了，如果不加你会得到  
一个"xxxx.h: No such file or directory"的错误。-I  
  
参数可以用相对路径，比如头文件在当前目录，可以用-I.来指定。上面我们提到的--cf  
lags参数就是用来生成-I参数的。  
  
6。-O参数  
这是一个程序优化参数，一般用-O2就是，用来优化程序用的，比如gcc test.c -O2，优  
化得到的程序比没优化的要小，执行速度可能也有所提  
  
高（我没有测试过）。  
  
7。-shared参数  
编译动态库时要用到，比如gcc -shared test.c -o libtest.so  
  
8。几个相关的环境变量  
PKG\_CONFIG\_PATH：用来指定pkg-config用到的pc文件的路径，默认是/usr/lib/pkgconf  
ig，pc文件是文本文件，扩展名是.pc，里面定义开发  
  
包的安装路径，Libs参数和Cflags参数等等。  
CC：用来指定c编译器。  
CXX：用来指定cxx编译器。  
LIBS：跟上面的--libs作用差不多。  
CFLAGS:跟上面的--cflags作用差不多。  
CC，CXX，LIBS，CFLAGS手动编译时一般用不上，在做configure时有时用到，一般情况  
下不用管。  
环境变量设定方法：export ENV\_NAME=xxxxxxxxxxxxxxxxx  
  
9。关于交叉编译  
交叉编译通俗地讲就是在一种平台上编译出能运行在体系结构不同的另一种平台上，比  
如在我们地PC平台(X86 CPU)上编译出能运行在sparc   
  
CPU平台上的程序，编译得到的程序在X86 CPU平台上是不能运行的，必须放到sparc   
CPU平台上才能运行。  
当然两个平台用的都是linux。  
  
这种方法在异平台移植和嵌入式开发时用得非常普遍。  
  
相对与交叉编译，我们平常做的编译就叫本地编译，也就是在当前平台编译，编译得到  
的程序也是在本地执行。  
  
用来编译这种程序的编译器就叫交叉编译器，相对来说，用来做本地编译的就叫本地编  
译器，一般用的都是gcc，但这种gcc跟本地的gcc编译器  
  
是不一样的，需要在编译gcc时用特定的configure参数才能得到支持交叉编译的gcc。  
  
为了不跟本地编译器混淆，交叉编译器的名字一般都有前缀，比如sparc-xxxx-linux-gn  
u-gcc，sparc-xxxx-linux-gnu-g++ 等等  
  
10。交叉编译器的使用方法  
使用方法跟本地的gcc差不多，但有一点特殊的是：必须用-L和-I参数指定编译器用spar  
c系统的库和头文件，不能用本地(X86)  
的库（头文件有时可以用本地的）。  
例子：  
sparc-xxxx-linux-gnu-gcc test.c -L/path/to/sparcLib -I/path/to/sparcInclude