<https://blog.csdn.net/u010977122/article/details/52993560>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

前几天在编译第三方开源库的时候，转载了[点击打开链接http://blog.csdn.net/u010977122/article/details/52958330](http://blog.csdn.net/u010977122/article/details/52958330)

这篇文章，里面详细介绍了linux系统中动态库和静态库的原理。通过这篇文章对对它们有了比较深的理解，后来交叉编译将交叉编译好的动态库拷贝到开发板上，不知道要放到哪里。开始以为是不是只要放在和可执行程序一起的目录就可以，可是程序执行的时候找不到，后来拷贝到/usr/lib 下，就可以执行了，感到很奇怪，程序是怎么找得到动态库。后来网上查阅，才知道是ldconfig 在其中“捣的鬼”。下面就网上搜到的介绍整理做了如下的记录。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**一、ldconfig**

ldconfig是一个动态链接库管理命令。为了让动态链接库为系统所共享,需运行动态链接库的管理命令--ldconfig。 ldconfig 命令的用途,主要是在默认搜寻目录(/lib和/usr/lib)以及动态库配置文件/etc/ld.so.conf内所列的目录下,搜索出可共享的动态链接库(格式lib\*.so\*),进而创建出动态装入程序(ld.so)所需的连接和缓存文件。缓存文件默认为 /etc/ld.so.cache,此文件保存已排好序的动态链接库名字列表，为了让动态链接库为系统所共享，需运行动态链接库的管理命令ldconfig，此执行程序存放在/sbin目录下。ldconfig通常在系统启动时运行,而当用户安装了一个新的动态链接库，修改了ld.so.conf时,就需要手工运行这个命令。

linux下的共享库机制采用了类似于高速缓存的机制，将库信息保存在/etc/ld.so.cache里边。程序连接的时候首先从这个文件里边查找，然后再到ld.so.conf的路径里边去详细找

**ldconfig命令行用法如下:**

ldconfig [-v|--verbose] [-n] [-N] [-X] [-f CONF] [-C CACHE] [-rROOT] [-l] [-p|--print-cache]

[-c FORMAT] [--format=FORMAT] [-V] [-?|--help|--usage] path...

**ldconfig可用的选项说明如下:**

(1) -v或--verbose : 用此选项时,ldconfig将显示正在扫描的目录及搜索到的动态链接库,还有它所创建的链接的名字.

(2) -n : 用此选项时,ldconfig仅扫描命令行指定的目录,不扫描默认目录(/lib,/usr/lib),也不扫描配置文件/etc/ld.so.conf所列的目录.

(3) -N : 此选项指示ldconfig不重建缓存文件(/etc/ld.so.cache).若未用-X选项,ldconfig照常更新文件的连接.

(4) -X : 此选项指示ldconfig不更新文件的连接.若未用-N选项,则缓存文件正常更新.

(5) -f CONF : 此选项指定动态链接库的配置文件为CONF,系统默认为/etc/ld.so.conf.

(6) -C CACHE : 此选项指定生成的缓存文件为CACHE,系统默认的是/etc/ld.so.cache,此文件存放已排好序的可共享的动态链接库的列表.

(7)  -r ROOT : 此选项改变应用程序的根目录为ROOT(是调用chroot函数实现的).选择此项时,系统默认的配置文件 /etc/ld.so.conf,实际对应的为 ROOT/etc/ld.so.conf.如用-r /usr/zzz时,打开配置文件 /etc/ld.so.conf时,实际打开的是/usr/zzz/etc/ld.so.conf文件.用此选项,可以大大增加动态链接库管理的灵活性.

(8) -l : 通常情况下,ldconfig搜索动态链接库时将自动建立动态链接库的连接.选择此项时,将进入专家模式,需要手工设置连接.一般用户不用此项.

(9) -p或--print-cache : 此选项指示ldconfig打印出当前缓存文件所保存的所有共享库的名字.

(10) -c FORMAT 或 --format=FORMAT : 此选项用于指定缓存文件所使用的格式,共有三种: ld(老格式),new(新格式)和compat(兼容格式,此为默认格式).

(11) -V : 此选项打印出ldconfig的版本信息,而后退出.

(12) -? 或 --help 或--usage : 这三个选项作用相同,都是让ldconfig打印出其帮助信息,而后退出.

**ldconfig几个需要注意的地方**  
1. 往/lib和/usr/lib里面加东西，是不用修改/etc/ld.so.conf的，但是完了之后要调一下ldconfig，不然这个library会找不到   
2. 想往上面两个目录以外加东西的时候，一定要修改/etc/ld.so.conf，然后再调用ldconfig，不然也会找不到   
比如安装了一个[MySQL](http://lib.csdn.net/base/14)到/usr/local/mysql，mysql有一大堆library在/usr/local/mysql/lib下面，这时 就需要在/etc/ld.so.conf下面加一行/usr/local/mysql/lib，保存过后ldconfig一下，新的library才能在 程序运行时被找到。   
3. 如果想在这两个目录以外放lib，但是又不想在/etc/ld.so.conf中加东西（或者是没有权限加东西）。那也可以，就是export一个全局变 量LD\_LIBRARY\_PATH，然后运行程序的时候就会去这个目录中找library。一般来讲这只是一种临时的解决方案，在没有权限或临时需要的时 候使用。   
4. ldconfig做的这些东西都与运行程序时有关，跟编译时一点关系都没有。编译的时候还是该加-L就得加，不要混淆了。   
5. 总之，就是不管做了什么关于library的变动后，最好都ldconfig一下，不然会出现一些意想不到的结果。不会花太多的时间，但是会省很多的事。

[**ldconfig提示“is not asymbolic link”的解决方法**](http://blog.csdn.net/choice_jj/article/details/8722035)

在编译的时候会出现以下错误：

ldconfig   
ldconfig: /lib/libdb-4.7.so is not a symbolic link

这是因为正常情况下libdb-4.7.so是一个符号连接，而不是一个实体文件，因此只需要把它改成符号连接即可

mv libdb-4.7.so libdb-4.so.7

ln -s libdb-4.so.7 libdb-4.7.so

**二、ldd**

作用：用来查看程序运行所需的共享库,常用来解决程序因缺少某个库文件而不能运行的一些问题。

语法：

ldd(选项)(参数)  
选项：

--version：打印指令版本号； -v：详细信息模式，打印所有相关信息； -u：打印未使用的直接依赖； -d：执行重定位和报告任何丢失的对象； -r：执行数据对象和函数的重定位，并且报告任何丢失的对象和函数； --help：显示帮助信息。  
参数：

文件：指定可执行程序或者文库。

ldd命令原理介绍：

1、首先ldd不是一个可执行程序，而只是一个shell脚本

2、ldd能够显示可执行模块的dependency，其原理是通过设置一系列的环境变量，如下：LD\_TRACE\_LOADED\_OBJECTS、LD\_WARN、LD\_BIND\_NOW、LD\_LIBRARY\_VERSION、LD\_VERBOSE等。当  
LD\_TRACE\_LOADED\_OBJECTS环境变量不为空时，任何可执行程序在运行时，它都会只显示模块的dependency，而程序并不真正执行。要不你可以在shell终端测试一下，如下：  
(1) export LD\_TRACE\_LOADED\_OBJECTS=1  
(2) 再执行任何的程序，如ls等，看看程序的运行结果

3、ldd显示可执行模块的dependency的工作原理，其实质是通过ld-linux.so（elf动态库的装载器）来实现的。我们知道，ld-linux.so模块会先于executable模块程序工作，并获得控制权，因此当上述的那些环境变量被设置时，ld-linux.so选择了显示可执行模块的dependency。

4、实际上可以直接执行ld-linux.so模块，如：/lib/ld-linux.so.2 --list program（这相当于ldd program）

向大家推荐一个linux下系统命令帮助查阅的网站，网站是中文的，拯救了一大波英文不好的，而且介绍很齐全，排版很简洁：<http://man.linuxde.net/>

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/u010977122/article/details/52993560