[linux下编译动态链接库的一个小例子](http://blog.csdn.net/qianchenglenger/article/details/32331129)

分类： [linux](http://blog.csdn.net/qianchenglenger/article/category/2380229)2014-06-19 19:44 1035人阅读 [评论](http://blog.csdn.net/qianchenglenger/article/details/32331129#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/qianchenglenger/article/details/32331129#report)

最近在项目中需要编译一个库，但是对linux下动态链接库的使用还不是很熟悉，所以，在此，写一个小的例子，制作自己的动态链接库，用该demo来熟悉一些linux下动态链接库的使用知识。

我们预备的demo的文件组成

1.testa.c

**[cpp]** [view plaincopy](http://blog.csdn.net/qianchenglenger/article/details/32331129)

1. #include <stdio.h>
2. **void** Test\_a()
3. {
4. printf("This is Test\_a!");
5. }

2. testb.c

**[cpp]** [view plaincopy](http://blog.csdn.net/qianchenglenger/article/details/32331129)

1. #include <stdio.h>
2. **void** Test\_b()
3. {
4. printf("This is Test\_b!");
5. }

3. testc.c

**[cpp]** [view plaincopy](http://blog.csdn.net/qianchenglenger/article/details/32331129)

1. #include <stdio.h>
2. **void** Test\_c()
3. {
4. printf("This is Test\_c!");
5. }

4. testh.h

**[cpp]** [view plaincopy](http://blog.csdn.net/qianchenglenger/article/details/32331129)

1. **void** Test\_a();
2. **void** Test\_b();
3. **void** Test\_c();

5. main.c

**[cpp]** [view plaincopy](http://blog.csdn.net/qianchenglenger/article/details/32331129)

1. #include "testh.h"
2. **int** main()
3. {
4. Test\_a();
5. Test\_b();
6. Test\_c();
7. **return** 0;
8. }

现在，我们将三个test函数编译为一个动态库，使用一下命令：

**[plain]** [view plaincopy](http://blog.csdn.net/qianchenglenger/article/details/32331129)

1. gcc -fPIC -shared testa.c testb.c testc.c -o libtest.so

生成了libtest.so文件，其中，命令中

-shared该选项指定生成动态连接库（让连接器生成T类型的导出符号表，有时候也生成弱连接W类型的导出符号），不用该标志外部程序无法连接。相当于一个可执行文件  
-fPIC：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进程的需要，而不能达到真正代码段共享的目的。

然后，通过通过命令

**[plain]** [view plaincopy](http://blog.csdn.net/qianchenglenger/article/details/32331129)

1. gcc main.c -L. -ltest -omain

编译生成应用程序main，其中命令中

-L.：表示要连接的库在当前目录中  
-ltest：编译器查找动态连接库时有隐含的命名规则，即在给出的名字前面加上lib，后面加上.so来确定库的名称

    我们知道一个程序要想在内存中运行，除了编译之外还要经过链接和装入这两个步骤。当然linux中动态链接也是经过这三个过程。Linux 使用这个ld-linux.so\*中的来装载（其实这只是一个链接）其他库。所以这个库必须放在linux中/lib下。对于其他，通常我们共享库放在/lib这个路径下，而且也是系统默认的搜索路径。  
Linux共享库的搜索路径先后顺序：  
1、编译目标代码时指定的动态库搜索路径：在编译的时候指定-Wl,-rpath=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

此处，我们使用了第2中方法来使用该动态库，先执行以下命令，设置LD\_LIBRARY\_PATH的值

**[plain]** [view plaincopy](http://blog.csdn.net/qianchenglenger/article/details/32331129)

1. export LD\_LIBRARY\_PATH=.

之后，使用一下命令即可

**[plain]** [view plaincopy](http://blog.csdn.net/qianchenglenger/article/details/32331129)

1. ./main

gcc shell.c -L. -lsqlite3 -lpthread -ldl -osqlite389