Linux操作系统下动态库的生成及链接方法 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/245/2021_2022_Linux_E6_93_8D_E4_BD_c103_245715.htm Linux下动态库文件的扩展名为".so"（Shared Object）。按照约定，所有动态库文件名的形式是libname.so（可能在名字中加入版本号）。这样，线程函数库被称作libthread.so。静态库的文件名形式是libname.a。共享archive的文件名形式是libname.sa。共享archive只是一种过渡形式，帮助人们从静态库转变到动态库。 本文仅以简单的例子介绍动态库文件的生成和链接方法。 操作系统：Debian/GNU Linux 2.6.21-2-686. GCC版本：4.1.3 一、库文件及测试文件代码 库文件及测试文件所在的目录：/home/program/。 1.库文件名：myfunction.c/*Author: Godbach E-mail: nylzhaowei@163.com*/#include stdio.h>intmy_lib_function (void){ printf ("Library routine called from libmyfunction.so!\n"). return 0.} 2.测试文件名：test.c#include stdio.h>int main(void){ my_lib_function(). return 0.} 二、动态库的编译方法 编译库文件myfunction.c:debian:/home/program# gcc -shared -o libmyfunction.so myfunction.c 如果编译成功，会在/home/program/下生成动态库文件：libmyfunction.so。 这里有两点需要说明： 1.对Linux操作，一般都推荐在普通用户模式下，如果需要超级用户的权限，则可以通过su root，输入root用户密码切换。我是个人学习使用，同时又有很多的操作都要使用root用户，因此就直接在root用户下进行编译。 2.编译生成动态库的命令为：gcc (-fpic) -shared -o

libmyfunction.so myfunction.c -fpic 使输出的对象模块是按照可重定位地址方式生成的。 -shared指定把对应的源文件生成对应的动态链接库文件。 三、动态库的测试方法 编译测试文件test.c:debian:/home/program# gcc -o test test.c /home/program/libmyfunction.so 成功编译后，生成test文件，运行test：debian:/home/program# ./test Library routine called from libmyfunction.so! 其中，gcc -o test test.c /home/program/libmyfunction.so的最后一个参数指定所链接库文件的绝对路径。本例中库文件的绝对路径为：/home/program/libmyfunction.so。 当然，如果想从系统的库文件路径（通常系统函数库的位于/usr/lib下）链接动态库的话，可以先将生成的库文件拷贝至/usr/lib/下，然后再链接：debian:/home/program# cp libmyfunction.so /usr/lib/debian:/home/program# gcc -o test test.c -lmyfunctiondebian:/home/program# ./test Library routine called from libmyfunction.so! 这里，对于链接的方法作一下解释。对于gcc -o test test.c -lmyfunction中最后一个参数-lmyfunction, 可见传给C编译器的命令行参数并未提到函数库的完整路径名，甚至没有提到在函数库目录中该文件的完整名字！实际上，编译器被告知根据选项-lmyfunction链接到相应的函数库(/usr/lib下)，函数库的名字是libmyfunction.so, 也就是说，"lib"部分和文件的扩展名被省略了，但在前面加了一个l。