在Linux中创建静态库和动态库 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/252/2021_2022__E5_9C_A8Linux_E4_B8_c103_252965.htm 我们通常把一些公用函数制作成函数库，供其它程序使用。函数库分为静态库和动态库两种。静态库在程序编译时会被连接到目标代码中，程序运行时将不再需要该静态库。动态库在程序编译时并不会被连接到目标代码中，而是在程序运行是才被载入，因此在程序运行时还需要动态库存在。本文主要通过举例来说明在Linux中如何创建静态库和动态库，以及使用它们。 在创建函数库前，我们先来准备举例用的源程序，并将函数库的源程序编译成.o文件。 第1步：编辑得到举例的程序--hello.h、hello.c和main.c； hello.c(见程序2)是函数库的源程序，其中包含公用函数hello，该函数将在屏幕上输出"Hello XXX!"。hello.h(见程序1)为该函数库的头文件。main.c(见程序3)为测试库文件的主程序，在主程序中调用了公用函数hello。 #ifndef HELLO_H #define HELLO_H void hello(const char *name). #endif //HELLO_H 程序1: hello.h #include void hello(const char *name) { printf("Hello %s!\n", name). } 程序2: hello.c #include "hello.h" int main() { hello("everyone"). return 0. } 程序3: main.c 第2步：将hello.c编译成.o文件； 无论静态库，还是动态库，都是由.o文件创建的。因此，我们必须将源程序hello.c通过gcc先编译成.o文件。 在系统提示符下键入以下命令得到hello.o文件。 # gcc -c hello.c # (注1：本文不介绍各命令使用和其参数功能，若希望详细了解它们，请参考其他文档。) (注2：首字符"#"是系统提示符，不需要键入，下文相同。) 我们运行ls命令看看

是否生存了hello.o文件。 # ls hello.c hello.h hello.o main.c # (注3：首字符不是"#"为系统运行结果，下文相同。) 在ls命令结果中，我们看到了hello.o文件，本步操作完成。 下面我们先来看看如何创建静态库，以及使用它。 第3步：由.o文件创建静态库； 静态库文件名的命名规范是以lib为前缀，紧接着跟静态库名，扩展名为.a。例如：我们将创建的静态库名为myhello，则静态库文件名就是libmyhello.a。在创建和使用静态库时，需要注意这点。创建静态库用ar命令。 在系统提示符下键入以下命令将创建静态库文件libmyhello.a。 # ar cr libmyhello.a hello.o # 我们同样运行ls命令查看结果： # ls hello.c hello.h hello.o libmyhello.a main.c # ls命令结果中有libmyhello.a。 第4步：在程序中使用静态库； 静态库制作完了，如何使用它内部的函数呢？只需要在使用到这些公用函数的源程序中包含这些公用函数的原型声明，然后在用gcc命令生成目标文件时指明静态库名，gcc将会从静态库中将公用函数连接到目标文件中。注意，gcc会在静态库名前加上前缀lib，然后追加扩展名.a得到的静态库文件名来查找静态库文件。 在程序3:main.c中，我们包含了静态库的头文件hello.h，然后在主程序main中直接调用公用函数hello。下面先生成目标程序hello，然后运行hello程序看看结果如何。 # gcc -o hello main.c -L. -lmyhello # ./hello Hello everyone! # 我们删除静态库文件试试公用函数hello是否真的连接到目标文件 hello中了。