Linux 下生成动态库和静态库

编辑得到举例的程序--hello.h、hello.c 和 main.c；

hello.h(见程序 1)为该函数库的头文件。

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!"。

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。

//hello.h

```
#ifndef HELLO_H
#define HELLO_H
   void hello(const char* name);
#endif
```

//hello.c

```
#include <stdio.h>
void hello(const char* name)
{
    printf("hello %s!\n", name);
}
```

//Main.c

```
#include "hello.h"
int main()
{
    hello("everyone");
    return 0;
}
```

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。

生成动态库：

```
root@mydebian:/home/michael/cppProject/libTest# gcc -shared -fpic -o libmyhello.so hello.c
root@mydebian:/home/michael/cppProject/libTest# gcc -o t main.c ./libmyhello.so
root@mydebian:/home/michael/cppProject/libTest# ./t
hello everyone!
```

-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。

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：

```
root@mydebian:/home/michael/cppProject/libTest# cp libmyhello.so /usr/lib/
root@mydebian:/home/michael/cppProject/libTest# gcc -o t main.c -lmyhello
root@mydebian:/home/michael/cppProject/libTest# ./t
hello everyone!
```

这里，对于链接的方法作一下解释。对于 gcc -o t main.c -lmyhello 中最后一个参数 -lmyhello, 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-lmyhello 链接到相应的函数库(/usr/lib 下)，函数库的名字是 libmyhello.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。

生成静态库

```
root@mydebian:/home/michael/cppProject/libTest# gcc -c hello.c
root@mydebian:/home/michael/cppProject/libTest# ls
hello.c    hello.h    hello.o    main.c
root@mydebian:/home/michael/cppProject/libTest# ar -cr libmyhello.a hello.o
root@mydebian:/home/michael/cppProject/libTest# ls
hello.c    hello.h    hello.o    libmyhello.a    main.c
```

在上述命令中已经生成静态链接库 libmyhello.a，可通过以下命令进行调用：

```
root@mydebian:/home/michael/cppProject/libTest# gcc -o t main.c libmyhello.a
root@mydebian:/home/michael/cppProject/libTest# ./t
hello everyone!
```

若将 libmyhello.a 复制到/usr/lib 目录下，将命令换成 gcc –o t main.c –lmyhello 即可。

```
root@mydebian:/home/michael/cppProject/libTest# mv libmyhello.a /usr/lib/
root@mydebian:/home/michael/cppProject/libTest# ls
hello.c    hello.h    hello.o    main.c
root@mydebian:/home/michael/cppProject/libTest# gcc -o t main.c -lmyhello
root@mydebian:/home/michael/cppProject/libTest# ./t
hello everyone!
```

当删除该静态链接库后，程序依然可以正常运行：

```
root@mydebian:/home/michael/cppProject/libTest# ls
hello.c    hello.h    hello.o    libmyhello.a    main.c    t
root@mydebian:/home/michael/cppProject/libTest# rm libmyhello.a
root@mydebian:/home/michael/cppProject/libTest# ./t
hello everyone!
```