静态库的制作

1. 命令规则： lib + 库的名字 + .a
2. 制作步骤：①生成对应的.o文件 .c ---->.o -c ②再将生成的.o文件打包,使用ar静态库打包工具

ar rcs + 静态库的名字(libMytest.a) + 生成的所有的.o

1. 发布和使用静态库

①发布静态库

②头文件:main.c中有head.h头文件的引用

Include lib main.c() src(源文件)

----- head.h ------libMycalc.a ----包含include “head.h” ------ a.c b.c

将include文件下的头文件.h和lib文件下的.a静态库一起发给用户

用户使用的时候，只需要在程序中调用include中的头文件即可.

最后编译的时候

Gcc main.c lib/libMycalc.a -o myapp -I include 直接生成即可.

Gcc main.c -I include -L lib -l Mycalc -o myapp

-L指定的静态库的目录,-l 指定库的名称(库的名字要去掉lib),大L ，小L

可以使用nm libMycalc.a 来查看动态库里面的内容

打包的时候，根据需求打包到可执行文件中去....例如调用了某个函数，则把这个函数的.o文件打包到可执行文件中去..

静态库的优缺点：

1. 发布程序的时候，不需要提供对应的库
2. 应用程序在调用库的时候，加载速度比较快
3. 库被打包到应用程序中，导致库的体积会很大
4. 库如果发生了改变的话，还需要重新编译

动态库的制作

1. 命名规则：lib + 名字 + .so
2. 制作步骤

①生成与位置无关的代码（生成.o文件）

Gcc -fPIC -c \*.c -I ../include

程序运行的时候才会去加载动态库，而静态库在程序未执行的时候就已经加载了,所以，每次加载的时候位置都不同.....

②将.o打包成动态库

Gcc -shared -o libMycalc.so \*.o -I include

1. 发布和使用共享库

把lib的libMycalc.so和include的head.h给了用户

Gcc main.c lib/libMycalc.so -o myapp -I include 生成对应的myapp文件即可

Gcc main.c -L lib - l Mycalc -I include 生成

Ldd 可查看程序所依赖的所有的动态库

\*\*\*\*如果无法找到动态链接库的话，就把它拷贝到根目录下的lib目录下\*\*\*\*

解决动态库链接失败的问题：

Export LD\_LIBRAY\_PATH=../lib

动态库的优缺点：

1. 程序在执行的时候才会被动态链接器加载到内存中
2. 执行程序体积小
3. 更新比较方便，不需要重新编译程序(函数的接口没变)
4. 动态库在发布应用程序的时候，需要将动态库提供给用户
5. 动态库没有被打包到应用程序中，加载速度相对较慢