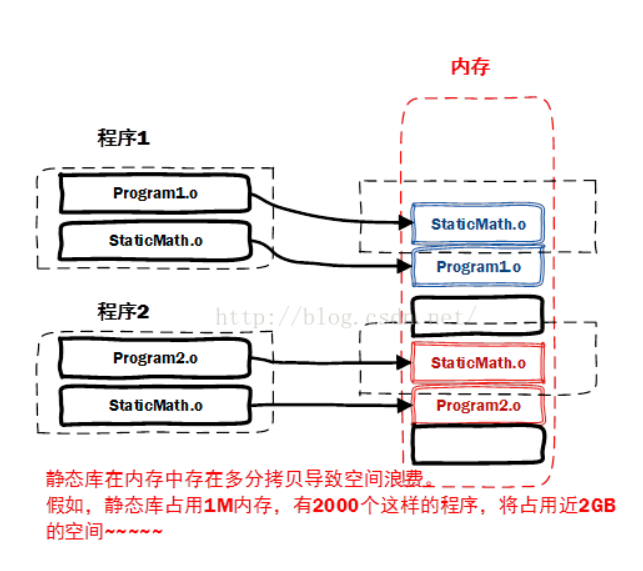
**静态链接库**

Win: name.lib

Linux: name.o

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。



以linux为例，创建静态库

1. 将代码编译成目标文件 gcc -c hello.c (-o hello.o)
2. 利用ar工具将目标文件打包成静态库文件

命名规范：**lib[your\_library\_name].a**

ar crv/rcs libmyhello.a hello.o 此处已经生成静态库文件

1. gcc -I (大写i) 指定头文件路径，gcc -I 指定头文件名字(一般不使用，而是直接放在\*\*.c 文件中通过#include<\*\*\*.h> 添加)
2. gcc -L 指定动态库或静态库路径
3. gcc -l (小写L)，指定链接库的名字，缺省优先动态库链接 libmyhello.a :-lmyhello，链接动态库：libmyhello.so : -lmyhello 注意：-l后面直接添加库名省去“lib”和“.so”或“.a”）
4. gcc -o main **main.c** -L/Users/wangxuewei/Desktop/静态链接 -lmyhello **生成可执行文件main**
5. 注意链接是在最后一步的，所以上述指令可拆分成两个

gcc -c main.c gcc main.o -L./ -lmyhello -o main

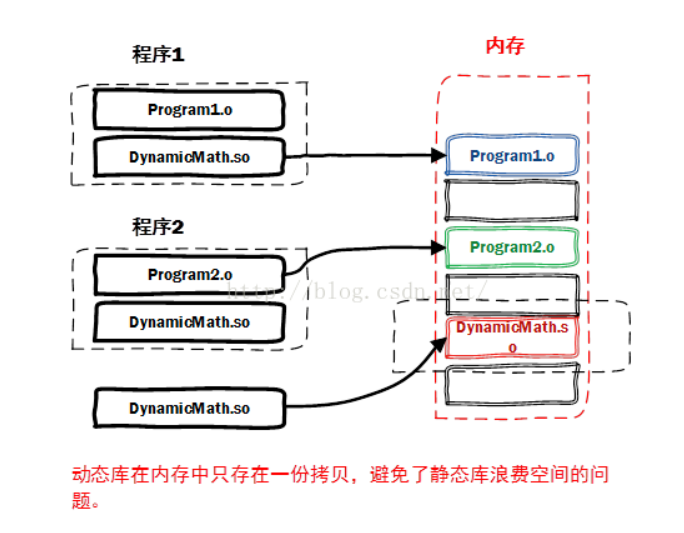
1. 如果路径下同时有静态库和动态库如何链接静态库，使用显示链接, gcc -l:libmyhello.a (将静态库的名字显示写出来)

**动态链接库**

Win: name.dll

Linux: name.so

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。



以linux为例，创建动态库

1. 生成目标文件，此时要加编译器选项-fPIC。[-fPIC是创建与地质无关的编译程序，是为了能够在多个应用程序间共享]

gcc -fPIC -c hello.c

1. 生成动态库，此时要加链接器选项-shared。[-shared是指定生成动态链接库]

gcc -shared -o libmyhello.so hello.o

1，2合并 gcc -shared -fPIC -o libmyhello.so hello.c

1. 链接动态库 gcc main.c -L. -lmyhello -o main **生成可执行文件main (-L. 代表动态库在当前路径下)**