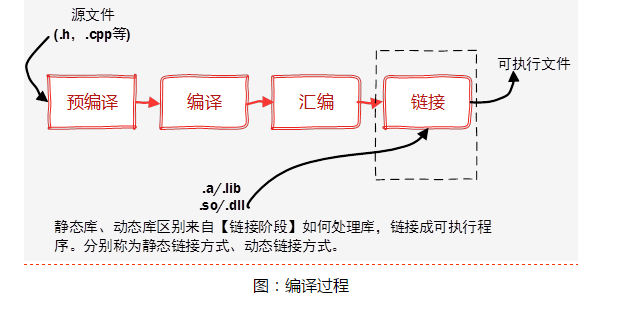
什么是库？

库是写好的现有的，成熟的，可以复用的代码。**现实中每个程序都要依赖很多基础的底层库，不可能每个人的代码都从零开始，因此库的存在意义非同寻常。**

本质上来说库是一种可执行代码的二进制形式，可以被操作系统载入内存执行。库有两种：静态库（.a、.lib）和动态库（.so、.dll）。

所谓静态、动态是指链接。回顾一下，将一个程序编译成可执行程序的步骤：



# **静态库：**

之所以成为【静态库】，是因为在链接阶段，会将汇编生成的目标文件.o与引用到的库一起链接打包到可执行文件中。因此对应的链接方式称为静态链接。

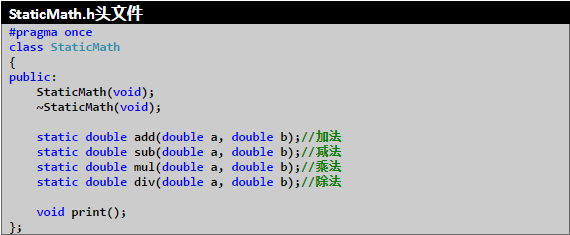
试想一下，静态库与汇编生成的目标文件一起链接为可执行文件，那么静态库必定跟.o文件格式相似。其实一个静态库可以简单看成是**一组目标文件（.o/.obj文件）的集合**，即很多目标文件经过压缩打包后形成的一个文件。静态库特点总结：

 l 静态库对函数库的链接是放在编译时期完成的。

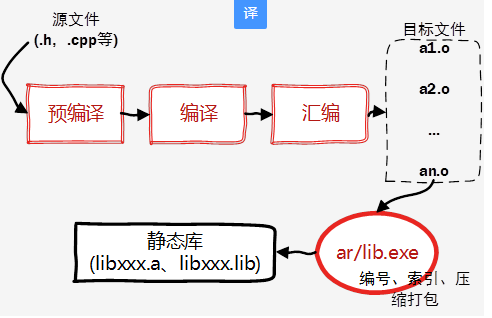
l  程序在运行时与函数库再无瓜葛，移植方便。

l  浪费空间和资源，因为所有相关的目标文件与牵涉到的函数库被链接合成一个可执行文件。

下面编写一些简单的四则运算C++类，将其编译成静态库给他人用，头文件如下所示：



Linux下使用***ar***工具、Windows下vs使用***lib.exe***，将目标文件压缩到一起，并且对其进行编号和索引，以便于查找和检索。一般创建静态库的步骤如图所示：



Linux静态库命名规范，必须是"lib[your\_library\_name].a"：lib为前缀，中间是静态库名，扩展名为.a。

### **创建静态库（.a）**

通过上面的流程可以知道，Linux创建静态库过程如下：

l  首先，将代码文件编译成目标文件.o（StaticMath.o）

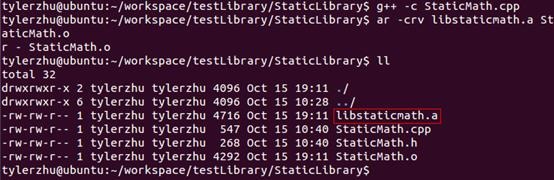
|  |
| --- |
| **g++ -c StaticMath.cpp** |

注意带参数-c，否则直接编译为可执行文件

l  然后，通过ar工具将目标文件打包成.a静态库文件

|  |
| --- |
| **ar -crv libstaticmath.a StaticMath.o** |

生成静态库***libstaticmath.a***。

[](http://images.cnitblog.com/blog/92071/201310/16201604-713c3a4a83aa4c17868a16f13b49027b.jpg)

大一点的项目会编写makefile文件（CMake等等工程管理工具）来生成静态库，输入多个命令太麻烦了。

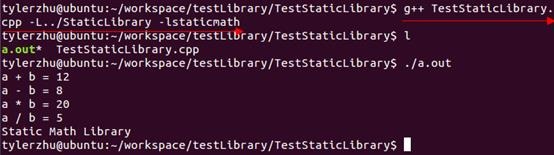
### **使用静态库**

编写使用上面创建的静态库的测试代码：

|  |
| --- |
| **测试代码：** |
| **#include "StaticMath.h"**  **#include <iostream>**  **using namespace std;**    **int main(int argc, char\* argv[])**  **{**  **double a = 10;**  **double b = 2;**    **cout << "a + b = " << StaticMath::add(a, b) << endl;**  **cout << "a - b = " << StaticMath::sub(a, b) << endl;**  **cout << "a \* b = " << StaticMath::mul(a, b) << endl;**  **cout << "a / b = " << StaticMath::div(a, b) << endl;**    **StaticMath sm;**  **sm.print();**    **system("pause");**  **return 0;**  **}** |

Linux下使用静态库，只需要在编译的时候，指定静态库的搜索路径（-L选项）、指定静态库名（不需要lib前缀和.a后缀，-l选项）。

# g++ TestStaticLibrary.cpp -**L../StaticLibrary** **-lstaticmath**

[](http://images.cnitblog.com/blog/92071/201310/16201604-ae57e89353c74265a0f1d26e89ffdbd5.jpg)

l  -L：表示要连接的库所在目录

l  -l：指定链接时需要的动态库，编译器查找动态连接库时有隐含的命名规则，即在给出的名字前面加上lib，后面加上.a或.so来确定库的名称。

## **Windows下创建与使用静态库**

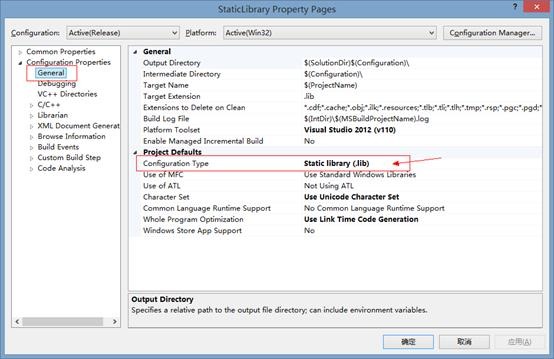
### **创建静态库（.lib）**

如果是使用VS命令行生成静态库，也是分两个步骤来生成程序：

l  首先，通过使用带编译器选项 **/c** 的 **Cl.exe** 编译代码 (**cl /c StaticMath.cpp**)，创建名为“StaticMath.obj”的目标文件。

l  然后，使用库管理器 **Lib.exe** 链接代码 (**lib StaticMath.obj**)，创建静态库StaticMath.lib。

当然，我们一般不这么用，使用VS工程设置更方便。创建win32控制台程序时，勾选静态库类型；打开工程“属性面板”è”配置属性”è”常规”，配置类型选择静态库。

[](http://images.cnitblog.com/blog/92071/201310/16201605-6a3acac10b40402ea99343ad092f0061.jpg)

图：vs静态库项目属性设置

Build项目即可生成静态库。

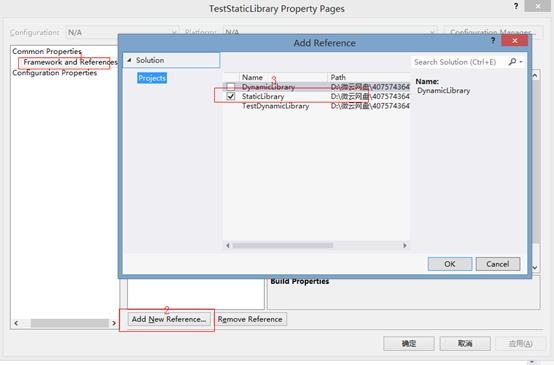
### **使用静态库**

测试代码Linux下面的一样。有3种使用方法：

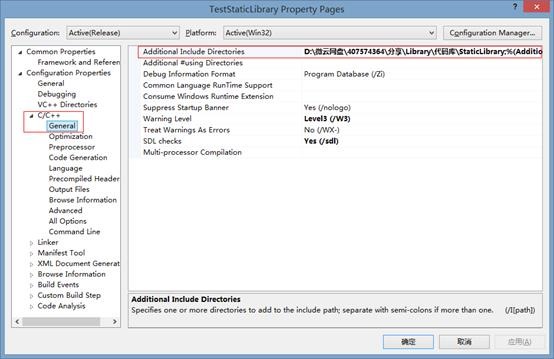
#### **方法一：**

在VS中使用静态库方法：

l  工程“属性面板”è“通用属性”è “框架和引用”è”添加引用”，将显示“添加引用”对话框。 “项目”选项卡列出了当前解决方案中的各个项目以及可以引用的所有库。 在“项目”选项卡中，选择 StaticLibrary。单击“确定”。

[](http://images.cnitblog.com/blog/92071/201310/16201607-fed082fd8a134770b24c2b4569ad4405.jpg)

l  添加StaticMath.h 头文件目录，必须修改包含目录路径。打开工程“属性面板”è”配置属性”è“C/C++”è” 常规”，在“附加包含目录”属性值中，键入StaticMath.h 头文件所在目录的路径或浏览至该目录。

[](http://images.cnitblog.com/blog/92071/201310/16201609-9be107869b604f53bc08653fe28bfbad.jpg)

编译运行OK。

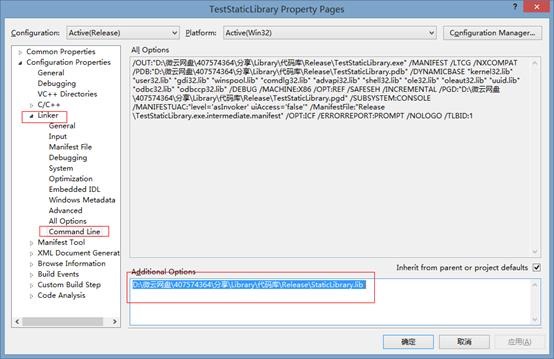
[](http://images.cnitblog.com/blog/92071/201310/16201610-8630c9a19cae4143bffacd255367ffbb.png)

图：静态库测试结果（vs）

如果引用的静态库不是在同一解决方案下的子工程，而是使用第三方提供的静态库lib和头文件，上面的方法设置不了。还有2中方法设置都可行。

#### **方法二：**

打开工程“属性面板”è”配置属性”è “链接器”è”命令行”，输入静态库的完整路径即可。

[](http://images.cnitblog.com/blog/92071/201310/16201610-2bde8b094f5a4603848c574f1f228144.jpg)

#### **方法三：**

l  “属性面板”è”配置属性”è “链接器”è”常规”，附加依赖库目录中输入，静态库所在目录；

l  “属性面板”è”配置属性”è “链接器”è”输入”，附加依赖库中输入静态库名StaticLibrary.lib。

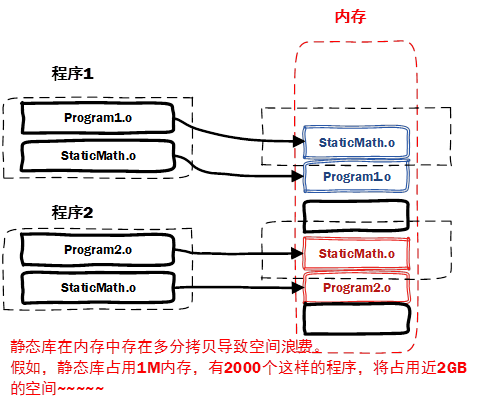
# **动态库**

通过上面的介绍发现静态库，容易使用和理解，也达到了代码复用的目的，那为什么还需要动态库呢？

## **为什么还需要动态库？**

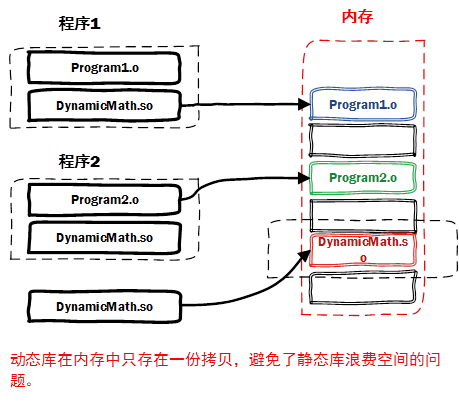
为什么需要动态库，其实也是静态库的特点导致。

l  空间浪费是静态库的一个问题。

[](http://images.cnitblog.com/blog/92071/201310/16201613-1320ec57feb24583a6b1294421c323b8.png)

l  另一个问题是静态库对程序的更新、部署和发布页会带来麻烦。如果静态库liba.lib更新了，所以使用它的应用程序都需要重新编译、发布给用户（对于玩家来说，可能是一个很小的改动，却导致整个程序重新下载，**全量更新**）。

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。**不同的应用程序如果调用相同的库，那么在内存里只需要有一份该共享库的实例**，规避了空间浪费问题。动态库在程序运行是才被载入，也解决了静态库对程序的更新、部署和发布页会带来麻烦。用户只需要更新动态库即可，**增量更新**。

[](http://images.cnitblog.com/blog/92071/201310/16201613-110ca9e0fd684281b3ee6d9bd9bebd78.png)

动态库特点总结：

l  动态库把对一些库函数的链接载入推迟到程序运行的时期。

l  可以实现进程之间的资源共享。（因此动态库也称为共享库）

l  将一些程序升级变得简单。

l  甚至可以真正做到链接载入完全由程序员在程序代码中控制（**显示调用**）。

Window与Linux执行文件格式不同，在创建动态库的时候有一些差异。

l  在Windows系统下的执行文件格式是PE格式，动态库需要一个**DllMain函数做出初始化的入口，通常在导出函数的声明时需要有\_declspec(dllexport)关键字**。

l  Linux下gcc编译的执行文件默认是ELF格式，**不需要初始化入口，亦不需要函数做特别的声明，**编写比较方便。

与创建静态库不同的是，不需要打包工具（ar、lib.exe），直接使用编译器即可创建动态库。