C++静态库与动态库

2013-10-16 20:18 by 吴秦, 77510 阅读, 33 评论, 收藏, 编辑

C++静态库与动态库

这次分享的**宗旨**是——让大家学会创建与使用静态库、动态库，知道静态库与动态库的区别，知道使用的时候如何选择。这里不深入介绍静态库、动态库的底层格式，内存布局等，有兴趣的同学，推荐一本书《程序员的自我修养——链接、装载与库》。

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：

About

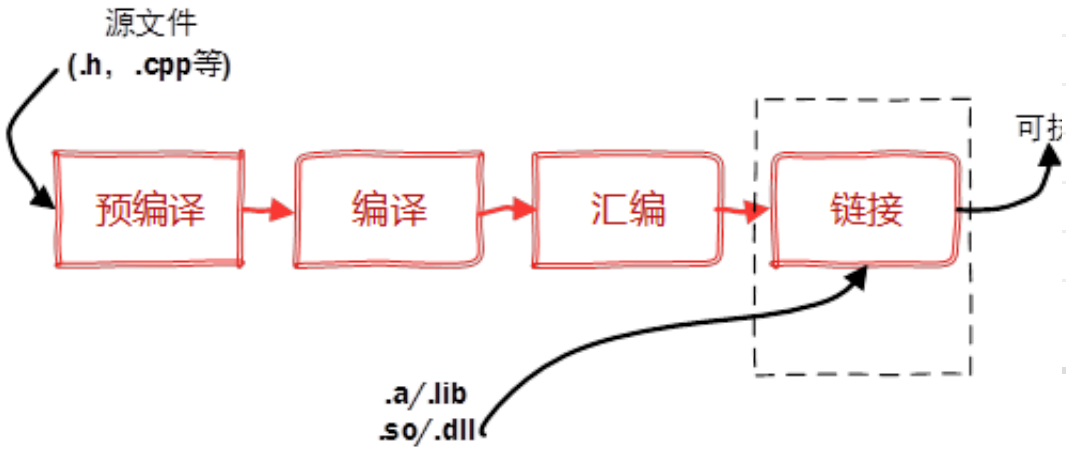
昵称：吴秦
园龄：7年1个月
荣誉：推荐博客
粉丝：3157
关注：18

+加关注

SEARCH

最新随笔

Unity3D手游开发实践
Unity3D shader简介
PyQt5应用与实践
Nginx + CGI/FastCGI + C/Cpp
Nginx安装与使用
优雅的使用Python之软件管理
优雅的使用python之环境管理
SpriteSheet精灵动画引擎
【译】AS3利用CPU缓存



图：编译过程

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：

- 静态库对函数库的链接是放在编译时期完成的。
- 程序在运行时与函数库再无瓜葛，移植方便。
- 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，

走在网页游戏开发的路上（十一）
自定义路径创建Cocos2d-x项目
C++静态库与动态库
C++对象模型
Python应用与实践
PureMVC（AS3）剖析：设计模式（二）

最新评论

Re:PyQt5应用与实践
膜拜大牛 -- 月海沐风
Re:C++静态库与动态库
打包成一个库，发布给其它的人用的时候，需要有头文件，头文件包含哪些接口可以供外界使用。 -- viola
Re:C++静态库与动态库
不论引用静态库还是动态库，都要在工程里面包含头文件吗？DynamicMath.h -- viola
Re:Nginx安装与使用
初学，很有帮助 -- chenxiaoqiong
Re:C++的函数重载
博主，我有个异议："Z后面的数字代表返回类型" 这个猜想我认为不妥。原因1:无论是C++还是Java，函数的重载都不以返回类型为依据。也就是如double foo() 和 int foo() 不能..... -- 代码钢琴家

日历

< 2016年11月 >

日	一	二	三	四	五	六
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19

随笔档案

2016年4月(1)
2015年8月(1)
2015年1月(1)
2014年12月(3)

头文件如下所示：

StaticMath.h头文件

```
#pragma once
class StaticMath
{
public:
    StaticMath(void);
    ~StaticMath(void);

    static double add(double a, double b);//加法
    static double sub(double a, double b);//减法
    static double mul(double a, double b);//乘法
    static double div(double a, double b);//除法

    void print();
};
```

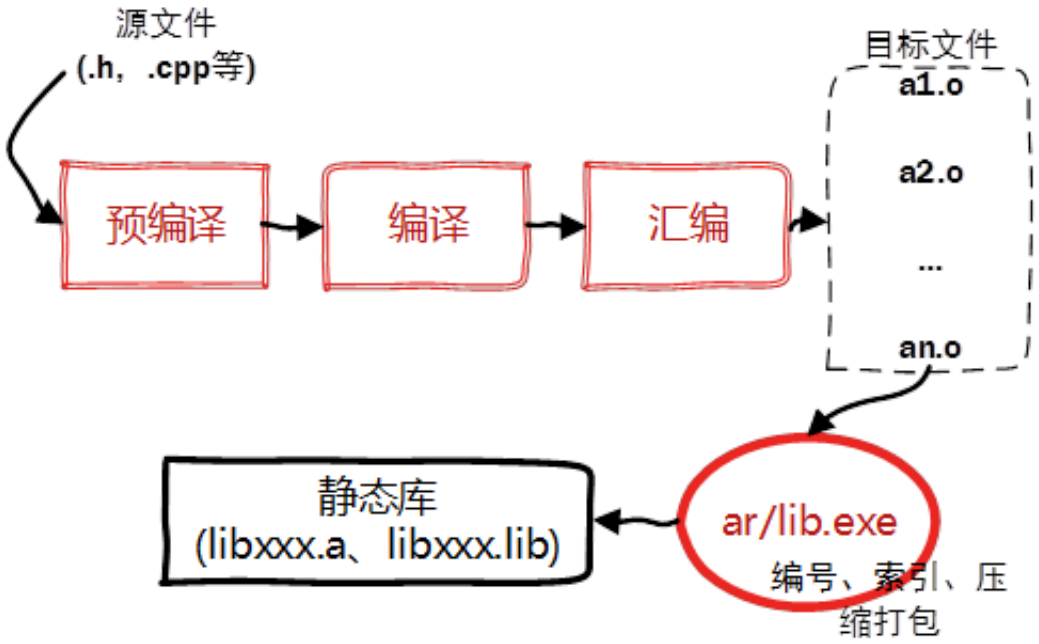
Linux下使用***ar***工具、Windows下vs使用***lib.exe***，将目标文件压缩到一起，并且对其进行编号和索引，以便于查找和检索。一般创建静态库的步骤如图所示：

20	21	22	23	24	25	26
27	28	29	30	1	2	3
4	5	6	7	8	9	10

随笔分类

.NET 2.0配置解谜系列(9)
.NET(C#) Internals (10)
【日常小记】(3)
【转载】(2)
Android开发之旅(18)
as3(1)
C/C++ Internals(15)
cocos2d-x(1)
JavaScript(1)
nginx(2)
PureMVC（AS3）剖析(5)
Python(5)
Unity3D(2)
Unix/Linux下编程(8)
服务器开发(3)
基于AIR Android应用开发(1)
客户端开发(1)
数据库(4)
网页游戏开发(23)
源码剖析：DotText源码学习(2)
源码剖析：Mongoose(5)

2014年11月(1)
2014年2月(3)
2013年11月(1)
2013年10月(1)
2013年9月(1)
2013年5月(1)
2013年3月(2)
2013年2月(2)
2013年1月(2)
2012年12月(4)
2012年11月(1)
2012年8月(1)
2012年4月(1)
2012年3月(2)
2012年1月(1)
2011年7月(1)
2011年6月(5)
2011年5月(3)
2011年3月(2)
2011年2月(1)
2011年1月(2)
2010年12月(6)
2010年10月(1)
2010年9月(4)
2010年7月(12)
2010年6月(4)
2010年5月(14)



图：创建静态库过程

Linux下创建与使用静态库

Linux静态库命名规则

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：

- 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

- 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
```

推荐排行榜

1. 字符集和字符编码（Charset & Encoding）(148)
2. Android开发之旅：环境搭建及HelloWorld(137)
3. HTTP协议及其POST与GET操作差异 & C#中如何使用POST、GET等(135)
4. Linux Socket编程（不限Linux）(118)
5. 浏览器缓存机制(82)
6. Unity3D手游开发实践(73)
7. C++静态库与动态库(62)
8. HTTP Keep-Alive模式(55)
9. Linux多线程编程（不限Linux）(53)
10. Android 开发之旅：view的几种布局方式及实践(45)

阅读排行榜

1. Android开发之旅：环境搭建及HelloWorld(1054601)
2. Nginx安装与使用(233134)
3. Linux Socket编程（不限Linux）(230537)
4. 字符集和字符编码（Charset & Encoding）(175732)
5. Android 开发之旅：view的几种布局方式及实践(98093)
6. Android开发之旅：android架构(97716)

2010年4月(12)
2010年3月(10)

生成静态库***libstaticmath.a***。

```
tylerzhu@ubuntu:~/workspace/testLibrary/StaticLibrary$ g++ -c StaticMath.cpp
tylerzhu@ubuntu:~/workspace/testLibrary/StaticLibrary$ ar -crv libstaticmath.a St
aticMath.o
r - StaticMath.o
tylerzhu@ubuntu:~/workspace/testLibrary/StaticLibrary$ ll
total 32
drwxrwxr-x 2 tylerzhu tylerzhu 4096 Oct 15 19:11 ./
drwxrwxr-x 6 tylerzhu tylerzhu 4096 Oct 15 10:28 ../
-rw-rw-r-- 1 tylerzhu tylerzhu 4716 Oct 15 19:11 libstaticmath.a
-rw-rw-r-- 1 tylerzhu tylerzhu  547 Oct 15 10:40 StaticMath.cpp
-rw-rw-r-- 1 tylerzhu tylerzhu  268 Oct 15 10:40 StaticMath.h
-rw-rw-r-- 1 tylerzhu tylerzhu 4292 Oct 15 19:11 StaticMath.o
tylerzhu@ubuntu:~/workspace/testLibrary/StaticLibrary$
```

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;
```

7. C++静态库与动态库(77510)
8. C++的函数重载(74298)
9. Android开发之旅：HelloWorld项目的目录结构(74088)
10. 【日常小记】linux中强大且常用命令：find、grep(73415)

系列索引帖

.NET 2.0配置解谜系列索引（完结）

```
}
```

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**

```
tylerzhu@ubuntu:~/workspace/testLibrary/TestStaticLibrary$ g++ TestStaticLibrary.
cpp -L../StaticLibrary -lstaticmath
tylerzhu@ubuntu:~/workspace/testLibrary/TestStaticLibrary$ l
a.out*  TestStaticLibrary.cpp
tylerzhu@ubuntu:~/workspace/testLibrary/TestStaticLibrary$ ./a.out
a + b = 12
a - b = 8
a * b = 20
a / b = 5
Static Math Library
tylerzhu@ubuntu:~/workspace/testLibrary/TestStaticLibrary$
```

- -L：表示要连接的库所在目录
- -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。

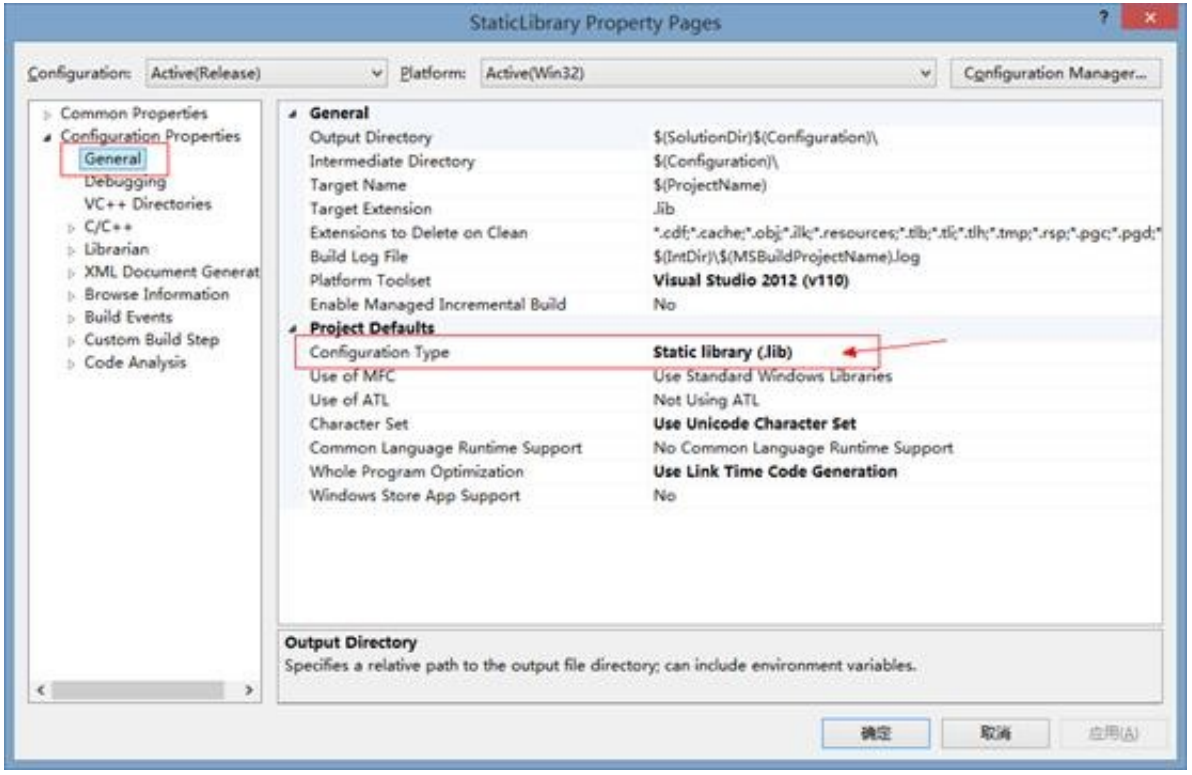
Windows下创建与使用静态库

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：

- 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。
- 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”➔”配置属性”➔”常规”，配置类型选择静态库。



图：vs静态库项目属性设置

Build项目即可生成静态库。

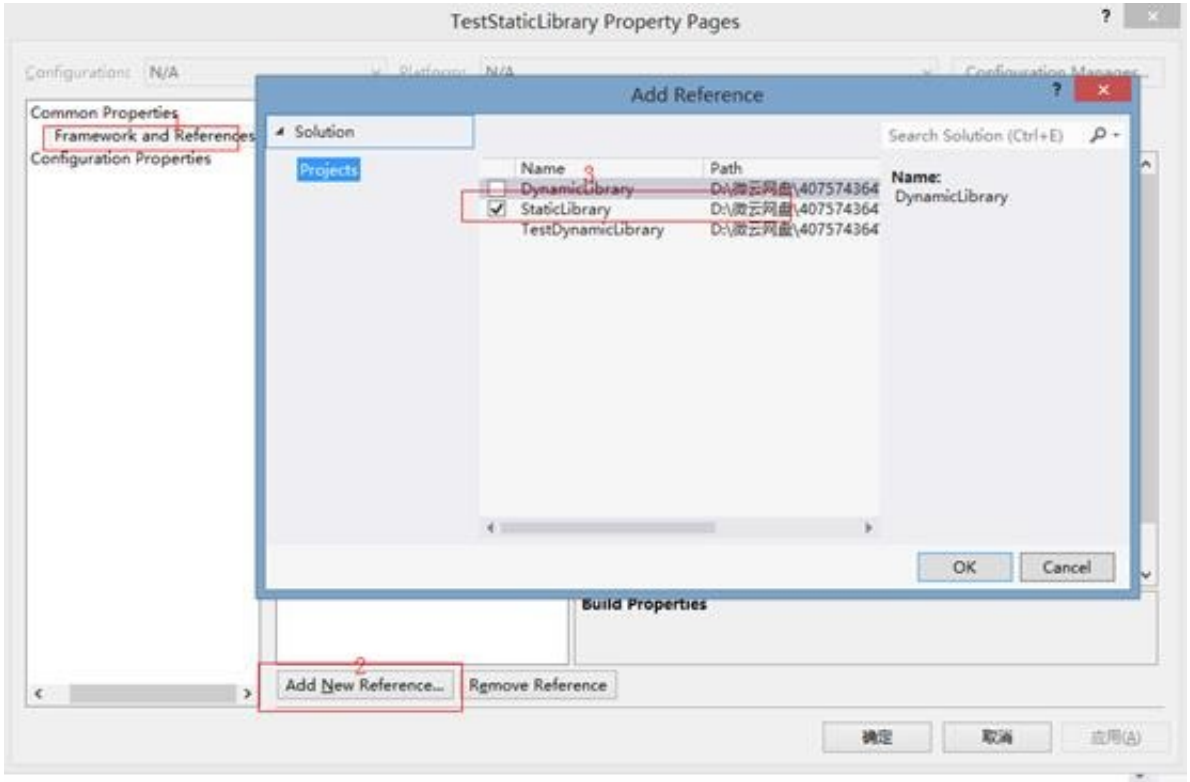
使用静态库

测试代码Linux下面的一样。有3种使用方法：

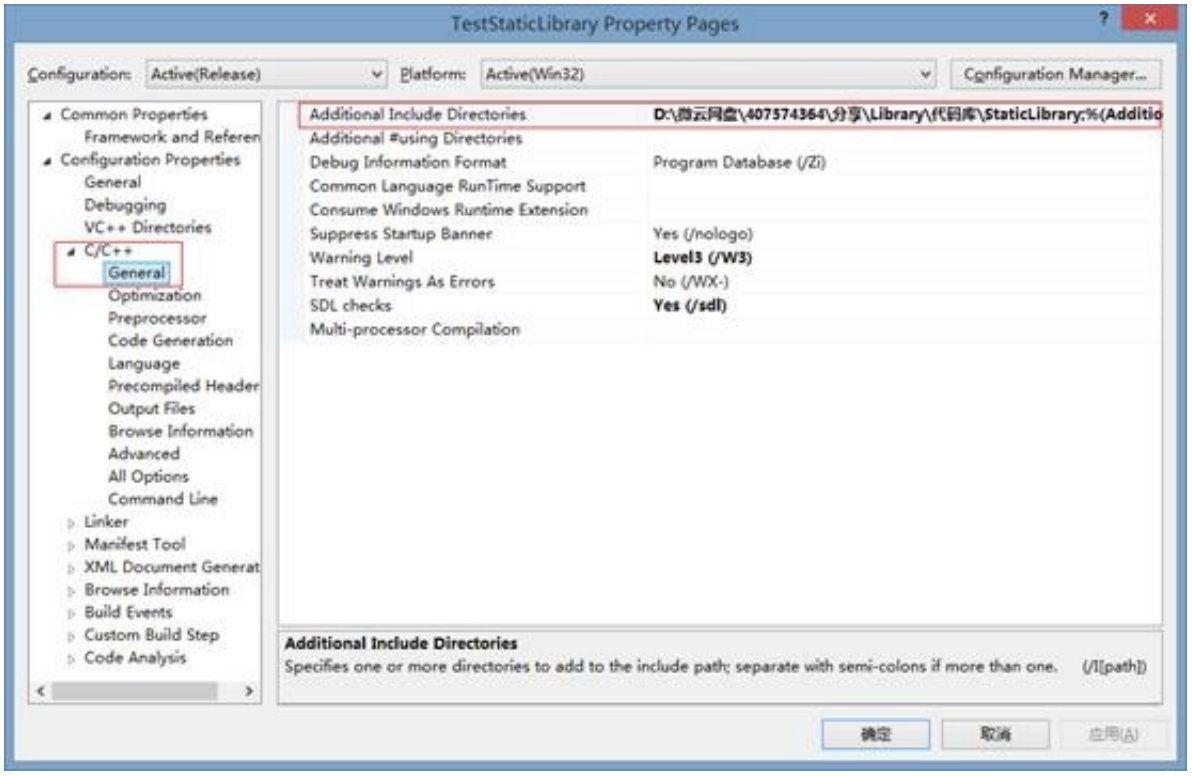
方法一：

在VS中使用静态库方法：

- 工程“属性面板”➔“通用属性”➔ “框架和引用”➔”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”。



- 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”➔”配置属性”➔ “C/C++”➔” 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。



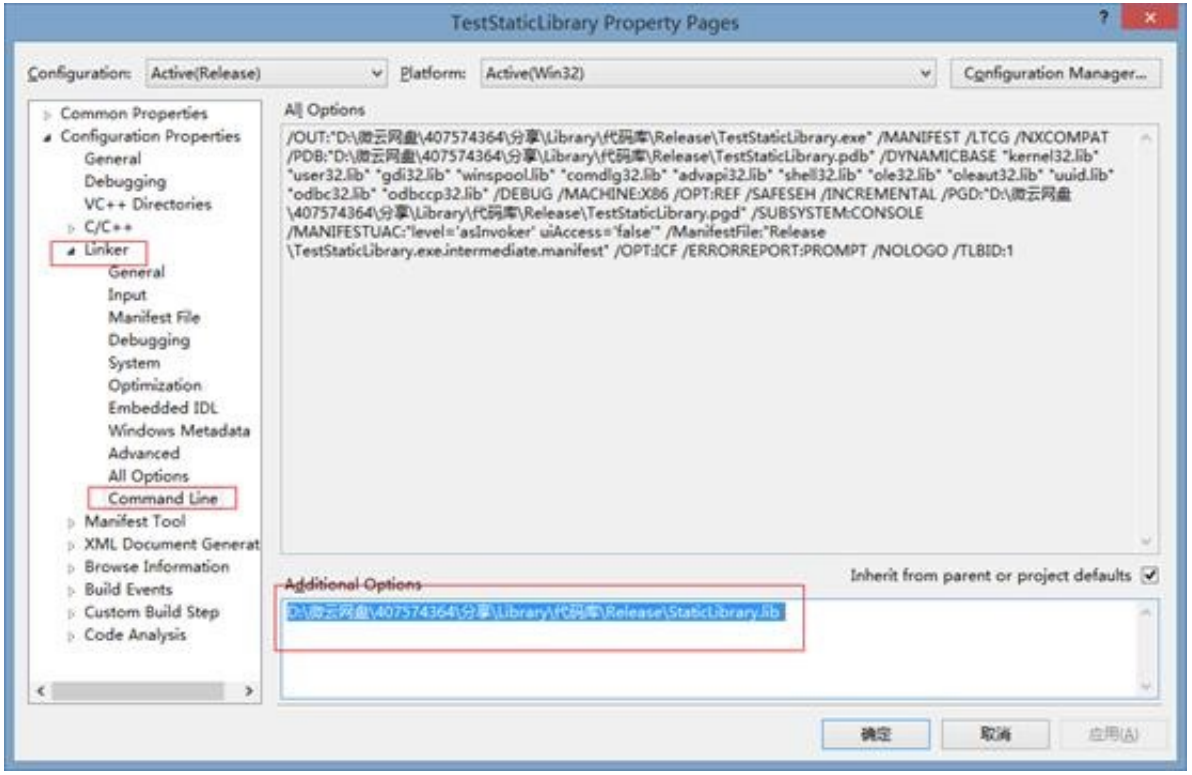
编译运行OK。

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。

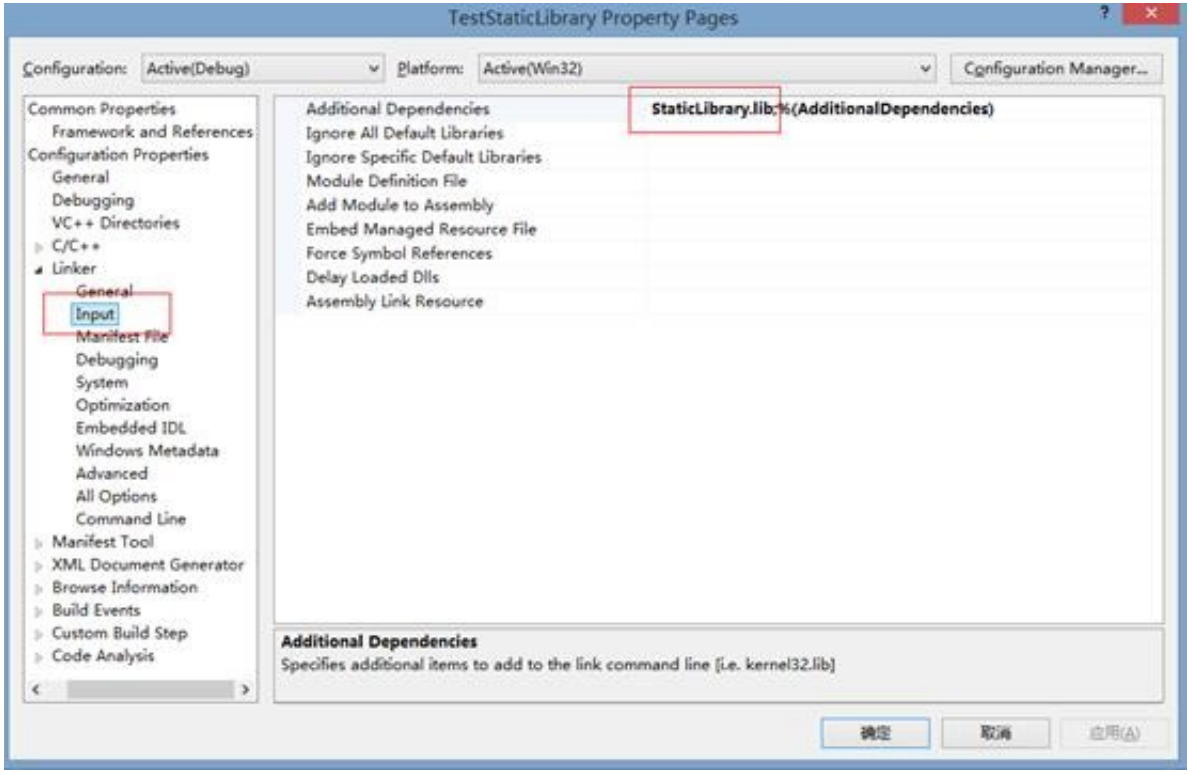
方法二：

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



方法三：

- “属性面板”➔”配置属性”➔ “链接器”➔”常规”，附加依赖库目录中输入，静态库所在目录；
- “属性面板”➔”配置属性”➔ “链接器”➔”输入”，附加依赖库中输入静态库名StaticLibrary.lib。



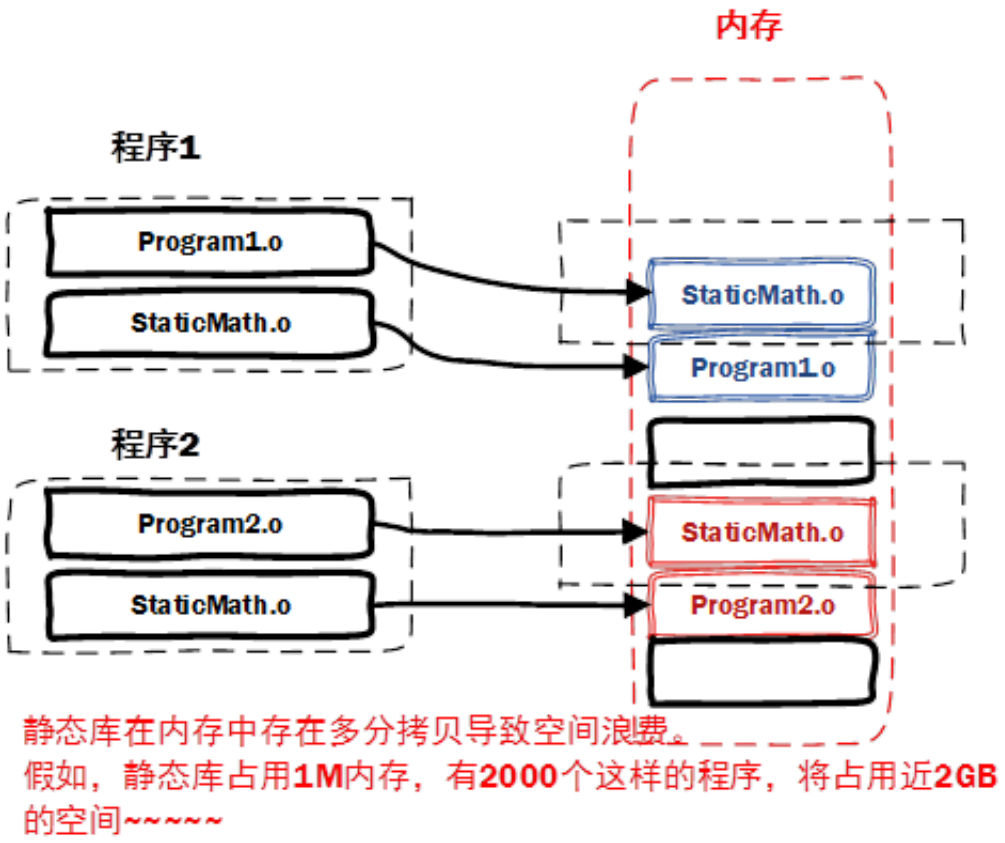
动态库

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？

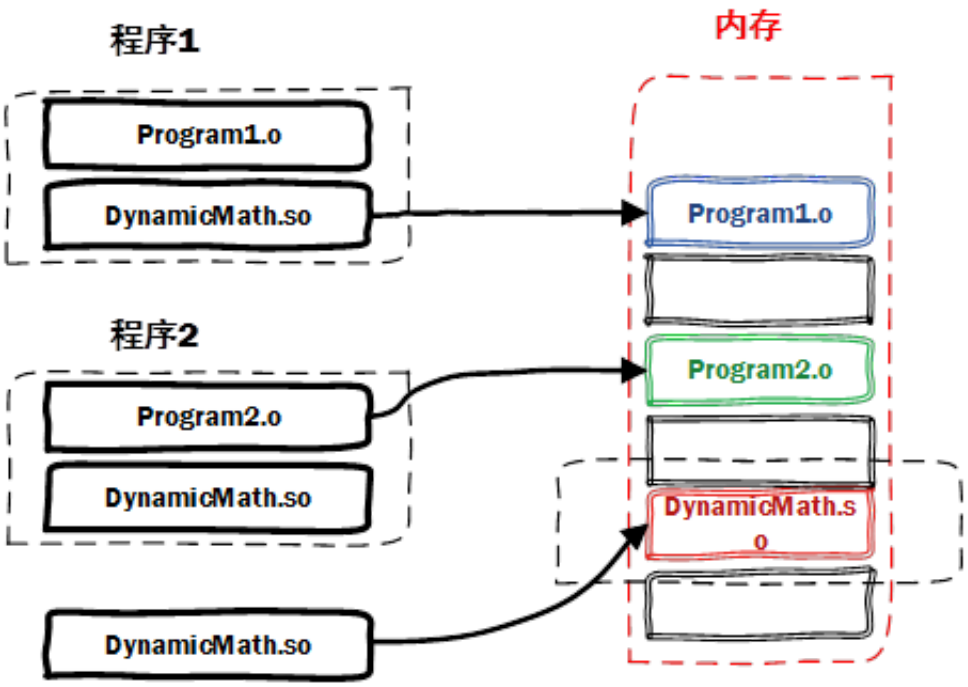
为什么需要动态库，其实也是静态库的特点导致。

- 空间浪费是静态库的一个问题。



- 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**。



动态库在内存中只存在一份拷贝，避免了静态库浪费空间的问题。

动态库特点总结：

- 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。
- 可以实现进程之间的资源共享。（因此动态库也称为共享库）
- 将一些程序升级变得简单。
- 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。

- 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。

- 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。

Linux下创建与使用动态库

linux动态库的命名规则

动态链接库的名字形式为 libxxx.so，前缀是lib，后缀名为“.so”。

- 针对于实际库文件，每个共享库都有个特殊的名字“soname”。在程序启动后，程序通过这个名字来告诉动态加载器该载入哪个共享库。
- 在文件系统中，soname仅是一个链接到实际动态库的链接。对于动态库而言，每个库实际上都有另一个名字给编译器来用。它是一个指向实际库镜像文件的链接文件（lib+soname+.so）。

创建动态库（.so）

编写四则运算动态库代码：

DynamicMath.h头文件

```
#pragma once
class DynamicMath
{
public:
	DynamicMath(void);
	~DynamicMath(void);

	static double add(double a, double b);//¼Ó·¨
	static double sub(double a, double b);//¼õ·¨
	static double mul(double a, double b);//³Ë·¨
```

```
    static double div(double a, double b);//³ý·¨
    void print();
};
```

- 首先，生成目标文件，此时要加编译器选项-fpic

```
g++ -fPIC -c DynamicMath.cpp
```

-fPIC 创建与地址无关的编译程序（pic，position independent code），是为了能够在多个应用程序间共享。

- 然后，生成动态库，此时要加链接器选项-shared

```
g++ -shared -o libdynmath.so DynamicMath.o
```

-shared指定生成动态链接库。

```
tylerzhu@ubuntu:~/workspace/testLibrary/DynamicLibrary$ g++ -fPIC -c DynamicMath.
cpp
tylerzhu@ubuntu:~/workspace/testLibrary/DynamicLibrary$ g++ -shared -o libdynmath
.so DynamicMath.o
tylerzhu@ubuntu:~/workspace/testLibrary/DynamicLibrary$ ll
total 28
drwxrwxr-x 2 tylerzhu tylerzhu 4096 Oct 15 22:32 ./
drwxrwxr-x 6 tylerzhu tylerzhu 4096 Oct 15 10:28 ../
-rw-rw-r-- 1 tylerzhu tylerzhu  503 Oct 15 21:45 DynamicMath.cpp
-rw-rw-r-- 1 tylerzhu tylerzhu  284 Oct 15 22:09 DynamicMath.h
-rw-rw-r-- 1 tylerzhu tylerzhu 3920 Oct 15 22:32 DynamicMath.o
-rwxrwxr-x 1 tylerzhu tylerzhu 8083 Oct 15 22:32 libdynmath.so*
tylerzhu@ubuntu:~/workspace/testLibrary/DynamicLibrary$
```

其实上面两个步骤可以合并为一个命令：

```
g++ -fPIC -shared -o libdynmath.so DynamicMath.cpp
```

使用动态库

编写使用动态库的测试代码：

```
测试代码：
#include "../DynamicLibrary/DynamicMath.h"
```

```
#include <iostream>
using namespace std;

int main(int argc, char* argv[])
{
    double a = 10;
    double b = 2;

    cout << "a + b = " << DynamicMath::add(a, b) << endl;
    cout << "a - b = " << DynamicMath::sub(a, b) << endl;
    cout << "a * b = " << DynamicMath::mul(a, b) << endl;
    cout << "a / b = " << DynamicMath::div(a, b) << endl;

    DynamicMath dyn;
    dyn.print();
    return 0;
}
```

引用动态库编译成可执行文件（跟静态库方式一样）：

```
g++ TestDynamicLibrary.cpp -L../DynamicLibrary -ldynmath
```

然后运行：./a.out，发现竟然报错了！！！

```
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$ ls
TestDynamicLibrary.cpp
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$ g++ TestDynamicLibrar
y.cpp -L../DynamicLibrary -ldynmath
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$ ls
a.out  TestDynamicLibrary.cpp
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$ ./a.out
./a.out: error while loading shared libraries: libdynmath.so: cannot open shared
object file: No such file or directory
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$
```

可能大家会猜测，是因为动态库跟测试程序不是一个目录，那我们验证下是否如此：

```
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$ cp ../DynamicLibrary/
libdynmath.so ./
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$ ls
a.out  libdynmath.so  TestDynamicLibrary.cpp
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$ ./a.out
./a.out: error while loading shared libraries: libdynmath.so: cannot open shared
object file: No such file or directory
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$
```

发现还是报错！！！那么，在执行的时候是如何定位共享库文件的呢？

1) 当系统加载可执行代码时候，能够知道其所依赖的库的名字，但是还需要知道绝对路径。此时就需要系统动态载入器(dynamic linker/loader)。

2) 对于elf格式的可执行程序，是由ld-linux.so*来完成的，它先后搜索elf文件的 DT_RPATH段—环境变量LD_LIBRARY_PATH —/etc/ld.so.cache文件列表—/lib/,/usr/lib 目录找到库文件后将其载入内存。

如何让系统能够找到它：

- 如果安装在/lib或者/usr/lib下，那么ld默认能够找到，无需其他操作。
- 如果安装在其他目录，需要将其添加到/etc/ld.so.cache文件中，步骤如下：
  - 编辑/etc/ld.so.conf文件，加入库文件所在目录的路径
  - 运行ldconfig ，该命令会重建/etc/ld.so.cache文件

我们将创建的动态库复制到/usr/lib下面，然后运行测试程序。

```
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$ sudo cp libdynmath.so
 /usr/lib/
[sudo] password for tylerzhu:
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$ ./a.out
a + b = 12
a - b = 8
a * b = 20
a / b = 5
```

Windows下创建与使用动态库

创建动态库（.dll）

与Linux相比，在Windows系统下创建动态库要稍微麻烦一些。首先，需要一个DllMain函数做出初始化的入口（创建win32控制台程序时，勾选DLL类型会自动生成这个文件）：

dllmain.cpp入口文件

```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "stdafx.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

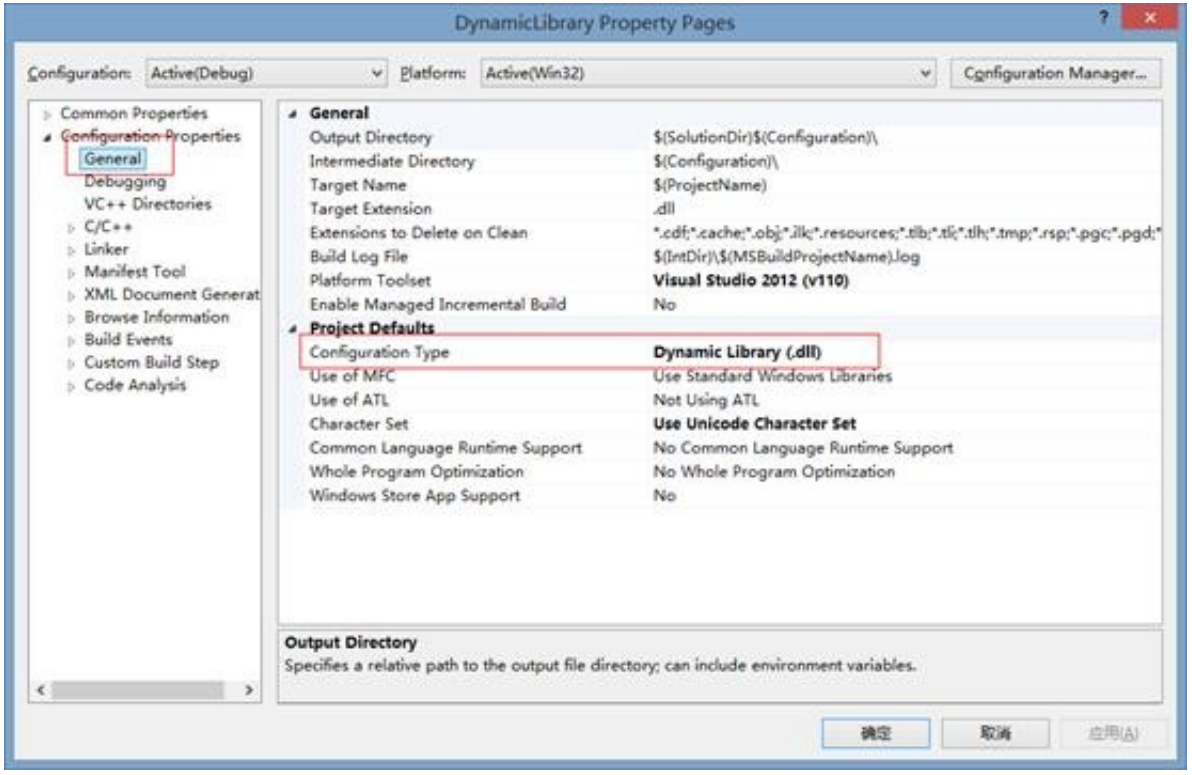
通常在导出函数的声明时需要有_declspec(dllexport)关键字：

DynamicMath.h头文件

```
#pragma once
```

```
class DynamicMath
{
public:
    __declspec(dllexport) DynamicMath(void);
    __declspec(dllexport) ~DynamicMath(void);

    static __declspec(dllexport) double add(double a, double
b);//加法
    static __declspec(dllexport) double sub(double a, double
b);//减法
    static __declspec(dllexport) double mul(double a, double
b);//乘法
    static __declspec(dllexport) double div(double a, double
b);//除法

    __declspec(dllexport) void print();
};
```

生成动态库需要设置工程属性，打开工程“属性面板”➔”配置属性”➔”常规”，配置类型选择动态库。



图：v动态库项目属性设置

Build项目即可生成动态库。

使用动态库

创建win32控制台测试程序：

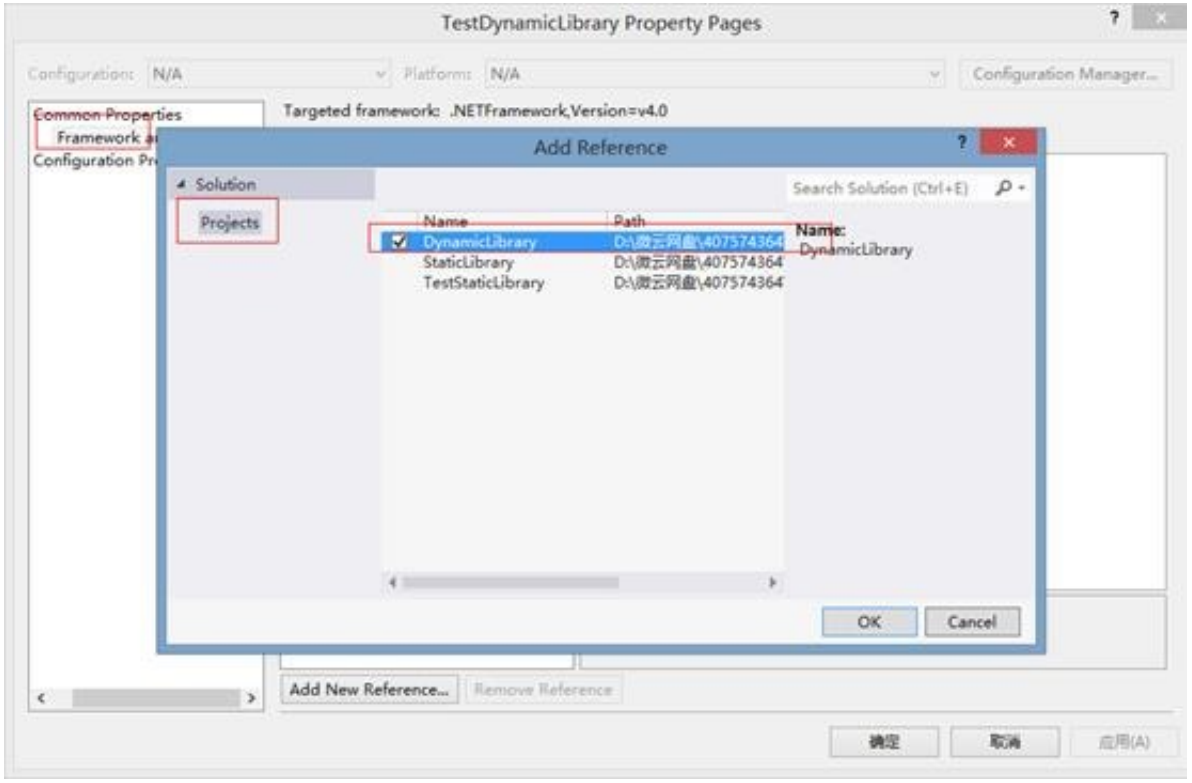
TestDynamicLibrary.cpp测试程序

```
#include "stdafx.h"
#include "DynamicMath.h"

#include <iostream>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    double a = 10;
    double b = 2;
```

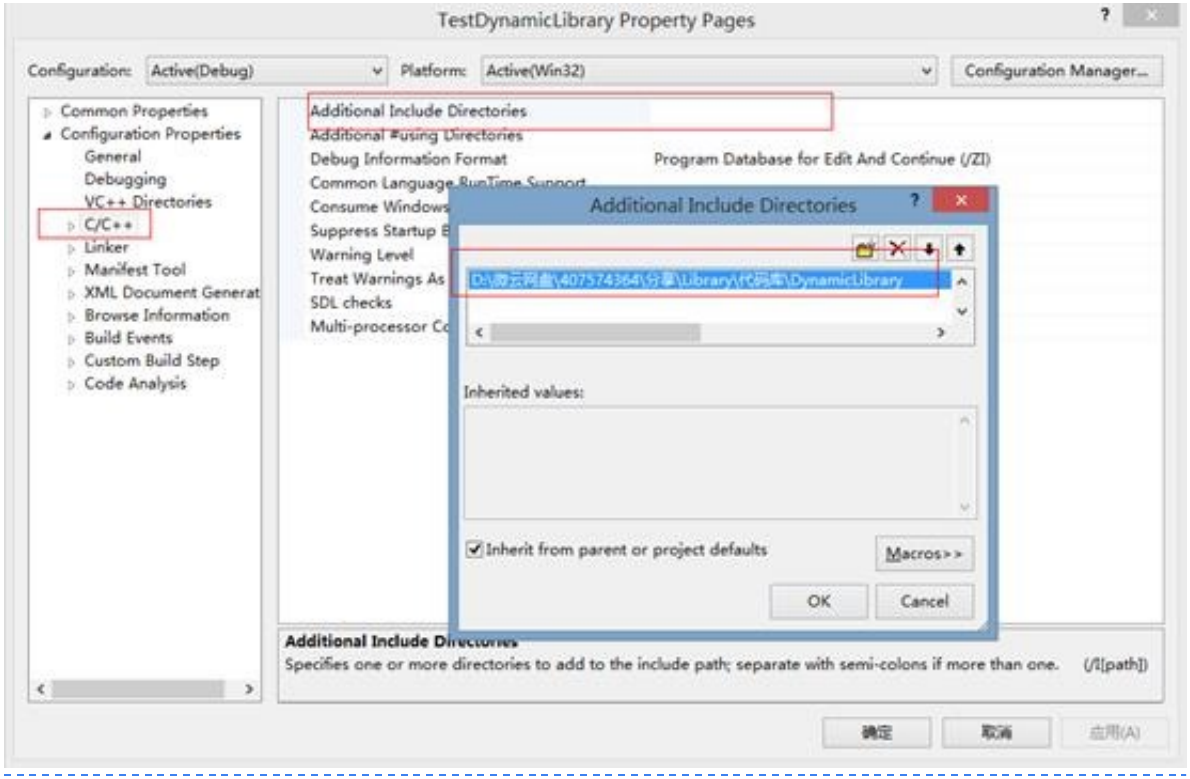
```
    cout << "a + b = " << DynamicMath::add(a, b) << endl;
    cout << "a - b = " << DynamicMath::sub(a, b) << endl;
    cout << "a * b = " << DynamicMath::mul(a, b) << endl;
    cout << "a / b = " << DynamicMath::div(a, b) << endl;

    DynamicMath dyn;
    dyn.print();

    system("pause");
    return 0;
}
```

方法一：

- 工程“属性面板”➔“通用属性”➔ “框架和引用”➔”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 DynamicLibrary。 单击“确定”。



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

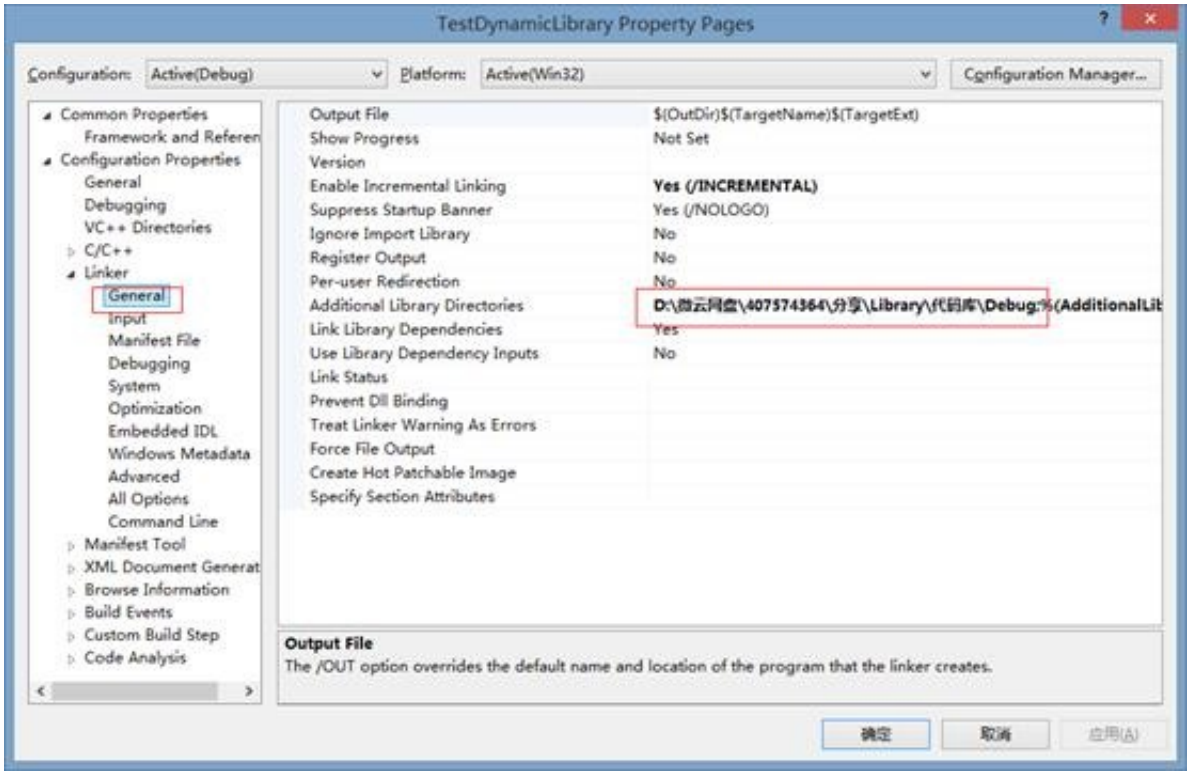


编译运行OK。

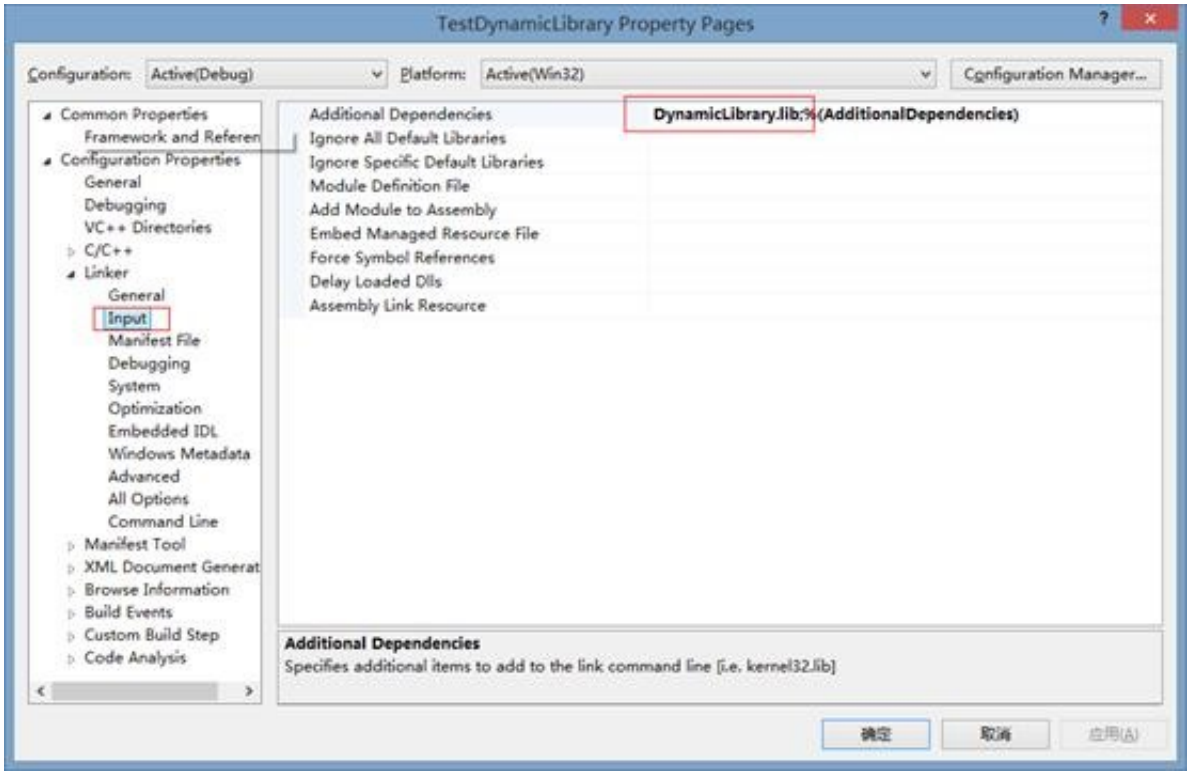
图：动态库测试结果（vs）

方法二：

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



- “属性面板”➔”配置属性”➔ “链接器”➔”输入”，附加依赖库中输入动态库编译出来的DynamicLibrary.lib。



这里可能大家有个疑问，动态库怎么还有一个DynamicLibrary.lib文件？即无论是静态链接库还是动态链接库，最后都有lib文件，那么两者区别是什么呢？其实，两个是完全不一样的东西。

DynamicLibrary.dll	2013/10/16 14:33	应用程序扩展	43 KB
DynamicLibrary.exp	2013/10/16 14:33	Exports Library ...	2 KB
DynamicLibrary.ilk	2013/10/16 14:33	Incremental Link...	234 KB
DynamicLibrary.lib	2013/10/16 14:33	Object File Library	3 KB
DynamicLibrary.pdb	2013/10/16 14:33	Program Debug...	963 KB
StaticLibrary.lib	2013/10/16 2:29	Object File Library	190 KB
TestDynamicLibrary.exe	2013/10/16 15:56	应用程序	65 KB
TestDynamicLibrary.ilk	2013/10/16 15:56	Incremental Link...	378 KB
TestDynamicLibrary.pdb	2013/10/16 15:56	Program Debug...	683 KB
TestStaticLibrary.exe	2013/10/16 14:33	应用程序	69 KB
TestStaticLibrary.ilk	2013/10/16 14:33	Incremental Link...	412 KB
TestStaticLibrary.pdb	2013/10/16 14:33	Program Debug...	899 KB

StaticLibrary.lib的大小为190KB，DynamicLibrary.lib的大小为3KB，静态库对应的lib文件叫**静态库**，动态库对应的lib文件叫**【导入库】**。实际上静态库本身就包含了实际执行代码、符号表等等，而**对于导入库而言，其实际的执行代码位于动态库中，导入库只包含了地址符号表等，确保程序找到对应函数的一些基本地址信息。**

动态库的显式调用

上面介绍的动态库使用方法和静态库类似属于隐式调用，编译的时候指定相应的库和查找路径。其实，动态库还可以显式调用。**【在C语言中】**，显示调用一个动态库轻而易举！

在Linux下显式调用动态库

#include <dlfcn.h>，提供了下面几个接口：

- void * **dlopen**(const char * pathname, int mode)：函数以指定模式打开指定的动态连接库文件，并返回一个句柄给调用进程。
- void* **dlsym**(void* handle,const char* symbol)：dlsym根据动态

链接库操作句柄(pHandle)与符号(symbol)，返回符号对应的地址。使用这个函数不但可以获取函数地址，也可以获取变量地址。

- int **dlclose** (void *handle)：dlclose用于关闭指定句柄的动态链接库，只有当此动态链接库的使用计数为0时,才会真正被系统卸载。
- const char *dlerror(void)：当动态链接库操作函数执行失败时，dlerror可以返回出错信息，返回值为NULL时表示操作函数执行成功。

在Windows下显式调用动态库

应用程序必须进行函数调用以在运行时显式加载 DLL。为显式链接到 DLL，应用程序必须：

- 调用 **LoadLibrary**（或相似的函数）以加载 DLL 和获取模块句柄。
- 调用 **GetProcAddress**，以获取指向应用程序要调用的每个导出函数的函数指针。由于应用程序是通过指针调用 DLL 的函数，编译器不生成外部引用，故无需与导入库链接。
- 使用完 DLL 后调用 **FreeLibrary**。

显式调用C++动态库注意点

对C++来说，情况稍微复杂。显式加载一个C++动态库的困难**一部分是因为C++的name mangling**；**另一部分是因为没有提供一个合适的API来装载类**，在C++中，您可能要用到库中的一个类，而这需要创建该类的一个实例，这不容易做到。

name mangling可以通过extern "C"解决。C++有个特定的关键字用来声明采用C binding的函数：extern "C" 。用 extern "C"声明的函数将使用函数名作符号名，就像C函数一样。因此，只有非成员函数才能被声明为extern "C"，并且不能被重载。尽管限制多多，extern "C"函数还是非常有用，因为它们可以象C函数一样被dlopen动态加载。冠以extern "C"限定符后，并不意味着函数中无法使用C++代码了，相反，

它仍然是一个完全的C++函数，可以使用任何C++特性和各种类型的参数。

另外如何从C++动态库中获取类，附上几篇相关文章，但我并不建议这么做：

- 《LoadLibrary调用DLL中的Class》：http://www.cppblog.com/codejie/archive/2009/09/24/97141.html
- 《C++ dlopen mini HOWTO》：http://blog.csdn.net/denny_233/article/details/7255673

“显式”使用C++动态库中的Class是非常繁琐和危险的事情，因此能用“隐式”就不要用“显式”，能静态就不要用动态。

附件：Linux下库相关命令

g++(gcc)编译选项

- -shared ：指定生成动态链接库。
- -static ：指定生成静态链接库。
- -fPIC ：表示编译为位置独立的代码，用于编译共享库。目标文件需要创建成位置无关码， 念上就是在可执行程序装载它们的时候，它们可以放在可执行程序的内存里的任何地方。
- -L. ：表示要连接的库所在的目录。
- -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/.so来确定库的名称。
- -Wall ：生成所有警告信息。
- -ggdb ：此选项将尽可能的生成gdb 的可以使用的调试信息。
- -g ：编译器在编译的时候产生调试信息。

- -c ：只激活预处理、编译和汇编,也就是把程序做成目标文件(.o文件) 。
- -Wl,options ：把参数(options)传递给链接器ld 。如果options 中间有逗号,就将options分成多个选项,然后传递给链接程序。

nm命令

有时候可能需要查看一个库中到底有哪些函数，**nm命令**可以打印出库中的涉及到的所有符号。库既可以是静态的也可以是动态的。nm列出的符号有很多，常见的有三种：

- 一种是在库中被调用，但并没有在库中定义(表明需要其他库支持)，用U表示；
- 一种是库中定义的函数，用T表示，这是最常见的；
- 一种是所谓的弱态”符号，它们虽然在库中被定义，但是可能被其他库中的同名符号覆盖，用W表示。

$nm libhello.h

ldd命令

ldd命令可以查看一个可执行程序依赖的共享库，例如我们编写的四则运算动态库依赖下面这些库：

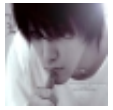
```
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$ ldd libdynmath.so
        linux-gate.so.1 =>  (0xb76f8000)
        libstdc++.so.6 => /usr/lib/i386-linux-gnu/libstdc++.so.6 (0xb75fe000)
        libgcc_s.so.1 => /lib/i386-linux-gnu/libgcc_s.so.1 (0xb75e0000)
        libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xb743a000)
        libm.so.6 => /lib/i386-linux-gnu/libm.so.6 (0xb740e000)
        /lib/ld-linux.so.2 (0xb76f9000)
tylerzhu@ubuntu:~/workspace/testLibrary/TestDynamicLibrary$
```

总结

二者的不同点在于**代码被载入的时刻不同。**

- 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，**因此体积较大。**
- 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，**因此代码体积较小。**

动态库的好处是，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。带来好处的同时，也会有问题！如经典的DLL Hell问题，关于如何规避动态库管理问题，可以自行查找相关资料。



« 上一篇：C++对象模型
» 下一篇：自定义路径创建Cocos2d-x项目

分类: C/C++ Internals,Unix/Linux下编程

ADD YOUR COMMENT

#1楼 john23.net
2013-10-17 09:28

感谢分享

支持(0) 反对(0)

#2楼 偶系小绵羊
2014-03-27 09:45

哥，我实在是找不到比你写的更清楚、更条理、更实用的技术性文章了，大赞一个！醍醐灌顶！！！

支持(9) 反对(0)

#3楼 叶秀标
2014-04-04 09:13

把晦涩的问题描述得简单易懂，很好的文章

支持(0) 反对(0)

#4楼 TYP
2014-04-07 22:33

写的很有水平，赞

支持(0) 反对(0)

#5楼 loverszhaokai
2014-05-06 07:27

请问楼主，博客中第一张图片是用什么工具做的？
谢谢！

支持(0) 反对(0)

#6楼[楼主] 吴秦
2014-05-07 08:56

@ loverszhaokai
visio 2013

支持(0) 反对(0)

#7楼 scutwang
2014-09-03 14:04

感谢楼主分享。

支持(0) 反对(0)

#8楼 奋斗+坚持
2014-11-06 15:08

感谢分享

支持(0) 反对(0)

#9楼 无冰蚂蚁
2014-11-06 16:02

写的非常不错，条理清晰，简单易懂。
但是
"显式"使用C++动态库中的Class是非常繁琐和危险的事情，因此能用"隐式"就不要用"显式"，能静态就不要用动态。这句话中能静态就不要用动态如何理解呢？
还有Windows下导入库的头文件不需要把 __declspec(dllexport) 换成 __declspec(dllimport)吗？

支持(0) 反对(0)

#10楼 neutral
2015-03-31 09:40

讲的太透彻了！

支持(0) 反对(0)

#11楼 我想吃个包子
2015-06-15 18:03

专门注册一个账号来给楼主点赞，这篇文章思路清晰，排版用心，我这种菜鸡看了都看懂了，找不到比这篇讲的更好的了。膜拜！

支持(1) 反对(0)

#12楼 hecangbo
2015-07-18 11:57

我在vs下做跟着做了静态库与动态库，完全能实现
我也是像楼上一样专门注册一个账号来给楼主点赞的
真的的讲解的很仔细，内容完整，一读就懂！

支持(1) 反对(0)

#13楼 、Dark
2015-08-10 21:24

使用动态库的时候需要头文件吗？动态库生成的.lib文件似乎可以替代头文件？

支持(0) 反对(0)

#14楼 xiongmao_cpp
2015-08-25 19:18

受教了，最近正好需要用到链接库，因此找到了这篇文章，现在对链接库有了一定的了解。感谢分享

支持(0) 反对(0)

#15楼 greed_a
2015-09-22 11:09

写的不催

支持(0) 反对(0)

#16楼 大众男神
2016-01-12 13:54

想请教一下作者用的是什么画图工具，觉得这个画图工具蛮好的，求推荐~

支持(0) 反对(0)

#17楼 木頭人の儛
2016-01-18 16:40

@ 大众男神

引用

想请教一下作者用的是什么画图工具，觉得这个画图工具蛮好的，求推荐~

同求~

支持(0) 反对(0)

#18楼 大众男神
2016-01-18 17:08

@ 朩頭人の僝
我找到了 是用Visio 2013画的，但是我没找到博主用的这个模板，你要是找到的话麻烦告诉我一下，谢谢~

支持(0) 反对(0)

#19楼 dazhou2016
2016-03-05 11:16

没错 我也是专门注册了一个账号来 感谢博主的 这比网上那些个垃圾博客要好他妈太多了 谢谢!!

支持(0) 反对(0)

#20楼 李小小白
2016-03-10 14:57

赞赞赞

支持(0) 反对(0)

#21楼 maddress2006
2016-04-07 13:49

赞赞赞

支持(0) 反对(0)

#22楼 QLTian
2016-04-20 09:46

作为一个新手，第一遍没读出什么味道。看了一点编译的知识，多读了几遍，收获了一些。仍需要继续努力，向博主致敬！

支持(0) 反对(0)

#23楼 stemon
2016-04-22 16:48

@ 大众男神
你确定这是用Visio画的吗？

支持(0) 反对(0)

#24楼[楼主] 吴秦
2016-04-23 12:13

@ stemon
的确是使用visio

支持(0) 反对(0)

#25楼 X东方晓X
2016-04-26 09:29

手动点赞，讲得透彻。

支持(0) 反对(0)

#26楼 李小小白
2016-05-06 17:33

windows 上创建动态库，可以不用WinMain 主函数吧，直接建立一个动态库的程序就可以了吧

支持(0) 反对(0)

#27楼 bl_stone
2016-05-18 11:25

讲的很好，感谢！

支持(0) 反对(0)

#28楼 jnqxhuanglei
2016-07-20 22:00

谢谢分享

支持(0) 反对(0)

#29楼 semiconlon
2016-09-09 22:41

很强

支持(0) 反对(0)

#30楼 西科小凡

2016-09-11 09:49

怒赞

支持(0) 反对(0)

#31楼 我是火车头

2016-10-20 14:21

深入浅出，赞

支持(0) 反对(0)

#32楼 viola

2016-10-27 17:28

不论引用静态库还是动态库，都要在工程里面包含头文件吗？DynamicMath.h

支持(0) 反对(0)

#33楼 viola

2016-10-28 10:03

打包成一个库，发布给其它的人用的时候，需要有头文件，头文件包含哪些接口可以供外界使用。

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

» 更多新闻...

最新知识库文章:

· 循序渐进地代码重构
· 技术的正宗与野路子
· 陈皓：什么是工程师文化？
· 没那么难，谈CSS的设计模式
· 程序猿媳妇儿注意事项
» 更多知识库文章...