# Boost教程

[Boost教程 1](#_Toc4564)

[boost简介 1](#_Toc16117)

[boost下载 1](#_Toc30891)

[Boost库外观 2](#_Toc25386)

[Boost的编译 3](#_Toc3064)

[在vc++中配置boost库 6](#_Toc2847)

[boost库不足之处 6](#_Toc18851)

[Boost常用库的使用介绍 6](#_Toc8220)

[Boost智能指针库 6](#_Toc21712)

[类库功能分类 8](#_Toc968)

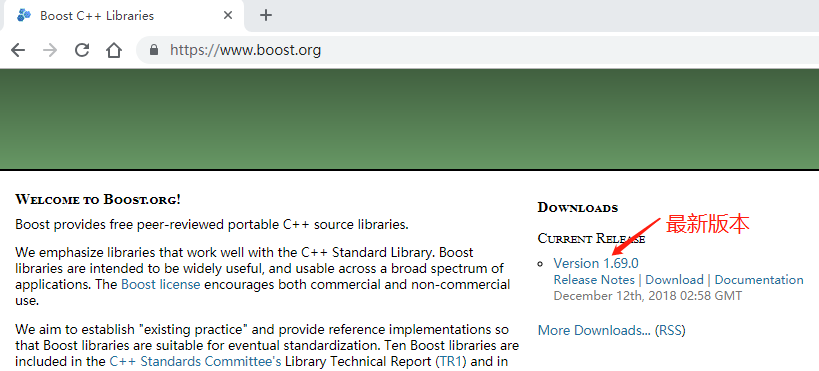
## boost简介

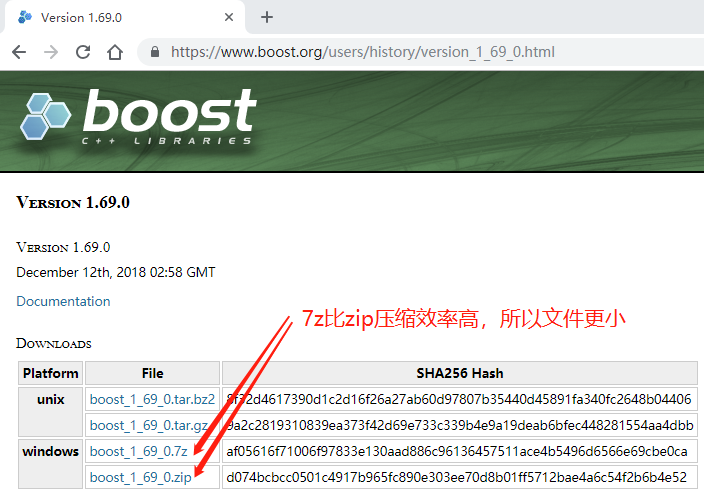
[Boost库](https://baike.baidu.com/item/Boost%E5%BA%93" \t "https://baike.baidu.com/item/boost/_blank)是一个可移植、提供[源代码](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81" \t "https://baike.baidu.com/item/boost/_blank)的C++库，作为标准库的后备，是C++标准化进程的开发引擎之一。 Boost库由C++标准委员会库工作组成员发起，其中有些内容有望成为下一代C++标准库内容。在C++社区中影响甚大，是不折不扣的“准”标准库。Boost由于其对跨平台的强调，对标准C++的强调，与编写平台无关。大部分boost库功能的使用只需包括相应头文件即可，少数（如[正则表达式](https://baike.baidu.com/item/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F" \t "https://baike.baidu.com/item/boost/_blank)库，文件系统库等）需要链接库。但Boost中也有很多是实验性质的东西，在实际的开发中使用需要谨慎。

Boost库是为C++语言标准库提供扩展的一些C++[程序库](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E5%BA%93/7662317" \t "https://baike.baidu.com/item/boost/_blank)的总称。

## boost下载

下载网址：<https://www.boost.org/>

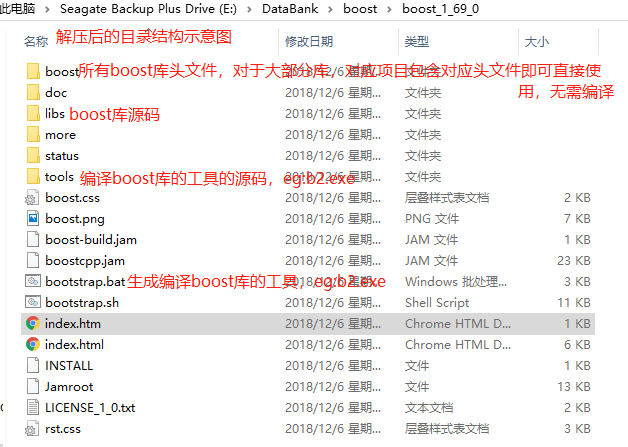




1549020242(1)

## Boost库外观

解压后的原始目录结构

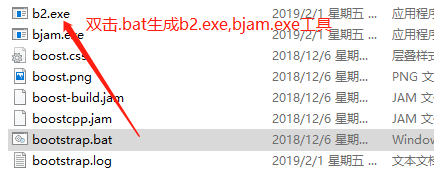


## Boost的编译

Boost库大部分类库是不需要编译，直接包含头文件即可

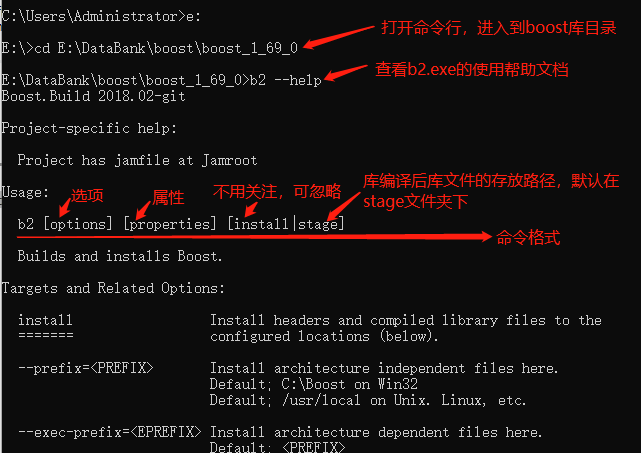
例如要使用share\_ptr,只需要在程序中#include”boost/shared\_ptr.hpp”

**1、生成boost库的编译工具**

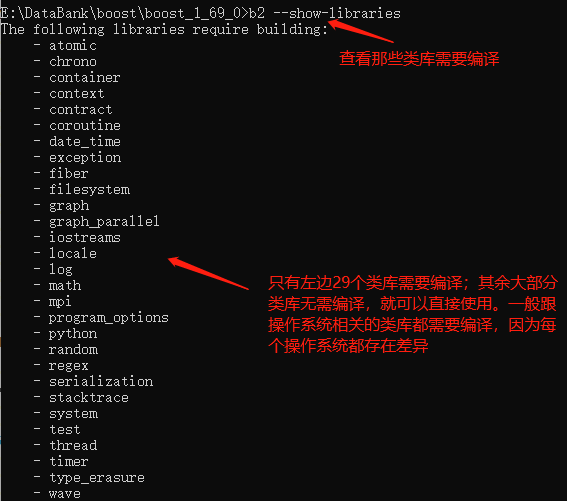


生成b2.exe和bjam.exe这两个应用程序，bjam.exe是早期编译boost库的应用程序，我们使用b2.exe来进行编译。

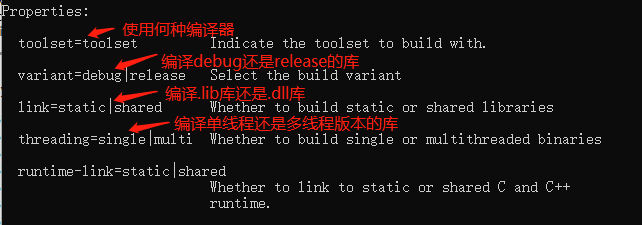
**b2.exe工具的使用：**



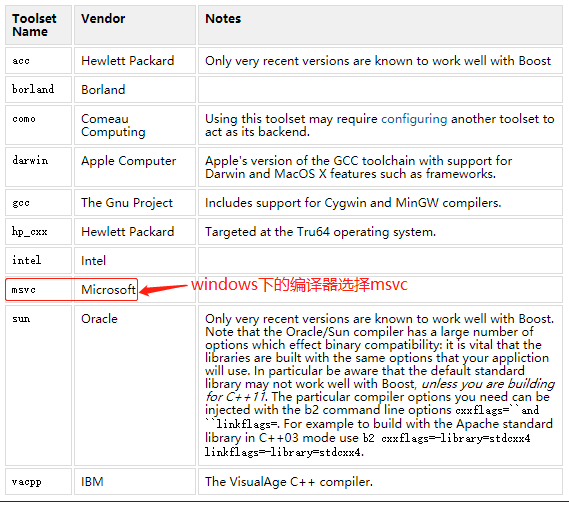
了解boost库所有需要编译的库的名称：b2 --show-libraries

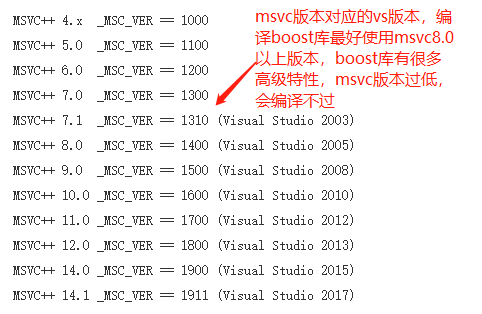


b2.exe的属性选项



**编译器的选择**





1. **编译boost库**

完全编译boost库：

b2 --toolset=msvc stage

上述命令执行后，经过差不多半小时的等待会将boost库编译成的所有debug和release版本的lib文件存放到stage/lib文件目录下。

局部编译仅用到的库：

Boost库比较庞大，我们不需要全部编译，用到哪些库就编译哪些库，例如使用with命令来编译data\_time库：

b2 --toolset-msvc-14.1 --with-data\_time stage

执行上述命令后，我们可以在stage/lib文件目录下发现刚才编译的库（debug和release静态库）

b2 --toolset=msvc-14.1 --without-graph --without-graph\_parallel --without-python --stagedir="F:\Development\_Tools\boost\boost\_1\_60\_0\lib" link=static runtime-link=static threading=multi address-model=64 debug release

说明：

--toolset：指定编译工具，VS2017的编译器版本号为：msvc-14.1

--without：不编译某项工具

--stagedir：指定编译后的库路径，不加这个描述默认在当前目录的stage\lib文件里面

## 在vc++中配置boost库

1、打开项目属性，选择头文件包含目录，加入E:\DataBank\boost\boost\_1\_69\_0（该目录为boost的根目录）

1、打开项目属性，选择库文件包含目录，加入E:\DataBank\boost\boost\_1\_69\_0\stage\lib

## boost库不足之处

日志库 (boost日志库还不够强大，可使用第三方库log4z)

更多脚本语言的支持 (目前仅支持python脚本)

数据库访问技术

垃圾回收器

Xml解析库

高级网络通讯协议库 (有asio库,但仅实现了tcp/ip,高层协议例如http/ftp协议还没有实现)

## Boost常用库的使用介绍

### Boost智能指针库

背景：C++程序员对内存管理的苦恼（内存泄漏、野指针、访问越界）

Boost的智能指针库就是内存管理的强劲解决方案

通过boost智能指针库，我们能够高效的进行内存管理，可以让你忘记栈（stack）,堆（heap）等内存相关的术语，并且还会发现，boost为c++提供的内存管理解决方案可能比java和c#等其它语言更好，更高效！

Boost提供下列几种智能指针

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ptr名称 | 获取管理权 | 共享管理权 | 转移管理权 | 拷贝赋值语意 | 重写比较操作符 | Hash函数 | 计数器 | 自动增减 |
| Shared\_ptr | y | y | y | y | y | y | y | y |
| Scoped\_ptr | y | n | y | n | n | n | n | n |
| Intrusive\_ptr | y | y | y | y | y | y | y | n |
| Weak\_ptr | n | n | n | y | y | y | n | n |
| Shared\_array | y | y | y | y | y | n | y | y |
| Scoped\_array | y | n | y | n | n | n | n | n |

**Shared\_ptr简介**

1. boost::Shared\_ptr是一个最像指针的”智能指针”

是boost.smart\_ptr中最有价值，最重要，也是最有用的类，boost库的其它许多类库都 是用了shared\_ptr,所以毫无悬念的被收入了c++11标准中去。

1. boost::Shared\_ptr实现了计数引用

它包装了new操作符在堆上分配的动态对象，但它实现了引用计数，可以自由的拷贝 和赋值，在任意地方共享它。当引用计数为0时，它会自动删除被包装的动态分配的对 象。

1. boost::Shared\_ptr不需要手动的调用类似release方法

它不像侵入式实现的智能指针一样需要手动的调用类似release方法，全部用由 shared\_ptr内部的计数器自动增减，这一点是非常有用的。（COM的IUnknow接口以 及boost::intrusive\_ptr<T>都是基于侵入式设计的智能指针，需要手动调用类似release 方法）

1. boost::Shared\_ptr支持所有权转移

并且可以安全的存储在stl标准容器中，是在stl容器存储指针的标准解法。例如 std::vector<int\*> IntVec,使用shared\_ptr方式为

std::vector<boost::shared\_ptr<int>> IntptrVec

**Weak\_ptr简介**

1. Weak\_ptr是用来解决循环引用和自引用对象

引用计数是一种很便利的内存管理机制，但是有一个很大的缺点，那就是不能管理循环 引用和自引用对象（例如链表和树节点），为了解决这个限制，因此weak\_ptr被引入 到boost的智能指针库中。

1. weak\_ptr并不能单独存在

它是与shared\_ptr同时使用的，它更像是shared\_ptr的助手而不是智能指针，因为它不 具备普通指针的行为，没有重载operator\*和->操作符，这是特意的。这样它就不能共 享指针，不能操作资源，这正是它“弱”的原因。它最大的作用是协助shared\_ptr工作， 像旁观者那样观察资源的使用情况。

1. weak\_ptr获得资源的观察权

Weak\_ptr可以从一个shared\_ptr或另外一个weak\_ptr构造，从而获得资源的观察权， 但weak\_ptr并没有共享资源，它的构造并不会引起引用计数的增加，同时它的析构也 不会引起引用计数的减少，他仅仅是观察者。

1. weak\_ptr可以被用于标准容器中的元素

Weak\_ptr实现了拷贝构造函数和重载了赋值操作符，因此weak\_ptr可以被用于标准容 器库中的元素，例如：在一个树节点中声明子树节点

std::vector<boost::weak\_ptr<node>> children;

**Shared\_ptr使用技巧**

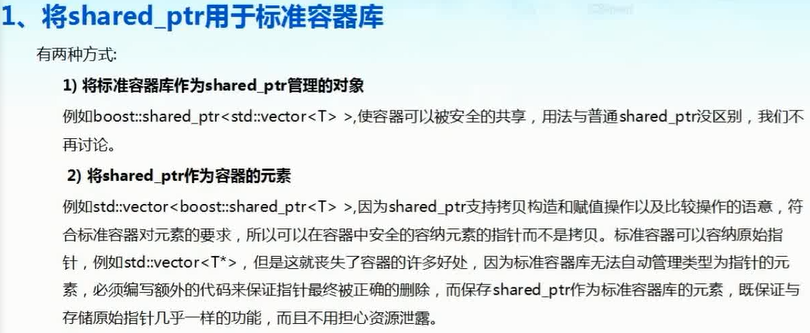
1. 将shared\_ptr用于标准容器库

有2种方式：

1. 、将标准容器库作为shared\_ptr管理的对象

例如boost::shared\_ptr<std::vector<T>>,使容器可以被安全的共享，用法与普通的 shared\_ptr没区别。

2）、将shared\_ptr作为容器的元素

****

## 类库功能分类

按照实现的功能，Boost可为大致归入以下20个分类，在下面的分类中，有些库同时归入几种类别。

**字符串和文本处理库**

a) Conversion库：对C++类型转换的增强，提供更强的[类型安全](https://baike.baidu.com/item/%E7%B1%BB%E5%9E%8B%E5%AE%89%E5%85%A8" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)转换、更高效的类型安全保护、进行范围检查的数值转换和词法转换。

b) Format库：实现类似printf的格式化对象，可以把参数格式化到一个字符串，而且是完全类型安全的。

c) IOStream库 ：扩展C++标准库流处理，建立一个流处理框架。

d) Lexical Cast库：用于字符串、整数、浮点数的字面转换。

e) Regex 库：[正则表达式](https://baike.baidu.com/item/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)，已经被TR1所接受。

f) Spirit库：基于EBNF范式的LL解析器框架

g) String Algo库：一组与字符串相关的算法

h) Tokenizer库：把字符串拆成一组记号的方法

i) Wave库：使用spirit库开发的一个完全符合C/C++标准的[预处理器](https://baike.baidu.com/item/%E9%A2%84%E5%A4%84%E7%90%86%E5%99%A8" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)

j) Xpressive 库：无需编译即可使用的[正则表达式](https://baike.baidu.com/item/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)库

**容器库**

a) Array 库：对C语言风格的数组进行包装

b) Bimap 库：双向映射结构库

c) Circular Buffer 库：实现循环缓冲区的数据结构

d) Disjoint Sets库 ：实现不相交集的库

e) Dynamic Bitset 库：支持运行时调整容器大小的位集合

f) GIL 库：通用图像库

g) Graph 库：处理图结构的库

h) ICL 库：区间容器库，处理区间集合和映射

i) Intrusive 库：侵入式容器和算法

j) Multi-Array 库：多维容器

k) Multi-Index 库：实现具有多个STL兼容索引的容器

l) Pointer Container 库：容纳[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)的容器

m) Property Map 库：提供键/值映射的属性概念定义

n) Property Tree 库：保存了多个属性值的树形数据结构

o) Unordered 库：散列容器，相当于hash\_xxx

p) Variant 库：简单地说，就是持有string, vector等复杂类型的[联合体](https://baike.baidu.com/item/%E8%81%94%E5%90%88%E4%BD%93" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)

**迭代器库**

a) GIL 库：通用图像库

b) Graph 库：处理图结构的库

c) Iterators 库：为创建新的[迭代器](https://baike.baidu.com/item/%E8%BF%AD%E4%BB%A3%E5%99%A8" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)提供框架

d) Operators 库：允许用户在自己的类里仅定义少量的操作符，就可方便地自动生成其他[操作符重载](https://baike.baidu.com/item/%E6%93%8D%E4%BD%9C%E7%AC%A6%E9%87%8D%E8%BD%BD" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)，而且保证正确的语义实现

e) Tokenizer 库：把字符串拆成一组记号的方法

**算法库**

a) Foreach库：容器遍历算法

b) GIL库：通用图像库

c) Graph库：处理图结构的库

d) Min-Max库：可在同一次操作中同时得到最大值和最小值

e) Range库：一组关于范围的概念和实用程序

f) String Algo库：可在不[使用正则表达式](https://baike.baidu.com/item/%E4%BD%BF%E7%94%A8%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)的情况下处理大多数字符串相关算法操作

g) Utility库：小工具的集合

**函数对象和高阶编程库**

a) Bind库：绑定器的泛化，已被收入TR1

b) Function库：实现一个通用的回调机制，已被收入TR1

c) Functional库：适配器的增强版本

d) Functional/Factory库：用于实现静态和动态的工厂模式

e) Functional/Forward库：用于接受任何类型的参数

f) Functional/Hash库：实现了TR1中的散列函数

g) Lambda库：Lambda表达式，即[未命名](https://baike.baidu.com/item/%E6%9C%AA%E5%91%BD%E5%90%8D" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)函数

h) Member Function库：是STL中mem\_fun和mem\_fun\_ref的扩展

i) Ref库：包装了对一个对象的引用，已被收入TR1

j) Result Of库：用于确定一个调用[表达式](https://baike.baidu.com/item/%E8%A1%A8%E8%BE%BE%E5%BC%8F" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)的返回类型，已被收入TR1

k) Signals库：实现[线程安全](https://baike.baidu.com/item/%E7%BA%BF%E7%A8%8B%E5%AE%89%E5%85%A8" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)的观察者模式

l) Signals2库：基于Signal的另一种实现

m) Utility库：小工具的集合

n) Phoenix库：实现在C++中的[函数式编程](https://baike.baidu.com/item/%E5%87%BD%E6%95%B0%E5%BC%8F%E7%BC%96%E7%A8%8B" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)。

**泛型编程库**

a) Call Traits库：封装可能是最好的函数传参方式

b) Concept Check库：用来检查是否符合某个概念

c) Enable If库:允许模板函数或[模板类](https://baike.baidu.com/item/%E6%A8%A1%E6%9D%BF%E7%B1%BB" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)在偏特化时仅针对某些特定类型有效

d) Function Types库：提供对函数、[函数指针](https://baike.baidu.com/item/%E5%87%BD%E6%95%B0%E6%8C%87%E9%92%88" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)、函数引用和[成员指针](https://baike.baidu.com/item/%E6%88%90%E5%91%98%E6%8C%87%E9%92%88" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)等类型进行分类分解和合成的功能

e) GIL库：通用图像库

f) In Place Factory, Typed In Place Factory库：工厂模式的一种实现

g) Operators库：允许用户在自己的类里仅定义少量的操作符，就可方便地自动生成其他操作符重载，而且保证正确的语义实现

h) Property Map库：提供键值映射的属性概念定义

i) Static Assert库：把断言的诊断时刻由运行期提前到编译期，让[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)检查可能发生的错误

j) Type Traits库：在编译时确定类型是否具有某些特征

k) TTI库：实现类型萃取的反射功能。

**模板元编程**

a) Fusion库：提供基于tuple的编译期容器和算法

b) MPL库：[模板元](https://baike.baidu.com/item/%E6%A8%A1%E6%9D%BF%E5%85%83" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)编程框架

c) Proto库：构建专用领域嵌入式语言

d) Static Assert库：把断言的诊断时刻由运行期提前到编译期，让编译器检查可能发生的错误

e) Type Traits库：在编译时确定类型是否具有某些特征

**预处理元编程库**

a) Preprocessors库：提供预处理元编程工具

**并发编程库**

a) Asio库：基于操作系统提供的异步机制，采用前摄设计模式实现了可移植的异步IO操作

b) Interprocess库：实现了可移植的[进程间通信](https://baike.baidu.com/item/%E8%BF%9B%E7%A8%8B%E9%97%B4%E9%80%9A%E4%BF%A1" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)功能，包括[共享内存](https://baike.baidu.com/item/%E5%85%B1%E4%BA%AB%E5%86%85%E5%AD%98" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)、[内存映射文件](https://baike.baidu.com/item/%E5%86%85%E5%AD%98%E6%98%A0%E5%B0%84%E6%96%87%E4%BB%B6" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)、[信号量](https://baike.baidu.com/item/%E4%BF%A1%E5%8F%B7%E9%87%8F" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)、[文件锁](https://baike.baidu.com/item/%E6%96%87%E4%BB%B6%E9%94%81" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)、[消息队列](https://baike.baidu.com/item/%E6%B6%88%E6%81%AF%E9%98%9F%E5%88%97" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)等

c) MPI库：用于高性能的分布式并行开发

d) Thread库：为C++增加线程处理能力，支持Windows和POSIX线程

e) Context库：提供了在单个线程上的协同式多任务处理的支持。该库可以用于实现用户级的多任务处理的机制，比如说协程coroutines，用户级协作线程或者类似于C#语言中yield关键字的实现。 [1]

f) Atomic库：实现[C++11](https://baike.baidu.com/item/C++11" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)样式的atomic<>，提供原子数据类型的支持和对这些原子类型的[原子操作](https://baike.baidu.com/item/%E5%8E%9F%E5%AD%90%E6%93%8D%E4%BD%9C" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)的支持。

g)Coroutine库：实现对[协程](https://baike.baidu.com/item/%E5%8D%8F%E7%A8%8B" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)的支持。协程与线程的不同之处在于，协程是基于合作式多任务的，而多线程是基于抢先式多任务的。

h)Lockfree库：提供对无锁数据结构的支持。

**数学和数字库**

a) Accumulators库：用于增量计算的累加器的框架

b) Integer库：提供一组有关整数处理的类

c) Interval库：处理区间概念的数学问题

d) Math库：数学领域的[模板类](https://baike.baidu.com/item/%E6%A8%A1%E6%9D%BF%E7%B1%BB" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)和算法

e) Math Common Factor库：用于支持最大公约数和最小公倍数

f) Math Octonion库 ：用于支持八元数

g) Math Quaternion库：用于支持四元数

h) Math/Special Functions库：数学上一些常用的函数

i) Math/Statistical Distributions库：用于单变量统计分布操作

j) Multi-Array库：多维容器

k) Numeric Conversion库：用于安全数字转换的一组函数

l) Operators库：允许用户在自己的类里仅定义少量的操作符，就可方便地自动生成其他操作符重载，而且保证正确的语义实现

m) Random库：专注于[伪随机数](https://baike.baidu.com/item/%E4%BC%AA%E9%9A%8F%E6%9C%BA%E6%95%B0" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)的实现，有多种算法可以产生高质量的伪随机数

n) Rational库：实现了没有精度损失的有理数

o) uBLAS库：用于线性代数领域的数学库

p) Geometry库：用于解决几何问题的概念、原语和算法。

q) Ratio库：根据C++ 0x标准N2661号建议 [2]  ，实现编译期的分数操作。

r)Multiprecision库：提供比C++内置的整数、分数和浮点数精度更高的多精度数值运算功能。 [3]

s)Odeint库：用于求解常微分方程的初值问题。 [4]

**排错和测试库**

a) Concept Check库 ：用来检查是否符合某个概念

b) Static Assert库 ：把断言的诊断时刻由运行期提前到编译期，让编译器检查可能发生的错误

c) Test库：提供了一个用于[单元测试](https://baike.baidu.com/item/%E5%8D%95%E5%85%83%E6%B5%8B%E8%AF%95" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)的基于[命令行界面](https://baike.baidu.com/item/%E5%91%BD%E4%BB%A4%E8%A1%8C%E7%95%8C%E9%9D%A2" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)的测试套件

**数据结构库**

a) Any库：支持对任意类型的值进行类型安全的存取

b) Bimap库：双向映射结构库

c) Compressed Pair库：优化的对pair对象的存储

d) Fusion库：提供基于tuple的编译期容器和算法

e) ICL库：区间容器库，处理区间集合和映射

f) Multi-Index库：为底层的容器提供多个索引

g) Pointer Container库：容纳指针的容器

h) Property Tree库：保存了多个属性值的树形数据结构

i) Tuple库：元组，已被TR1接受

j) Uuid库：用于表示和生成UUID

k) Variant库：有类别的[泛型](https://baike.baidu.com/item/%E6%B3%9B%E5%9E%8B" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)联合类

l) Heap库：对std::priority\_queue扩展，实现[优先级队列](https://baike.baidu.com/item/%E4%BC%98%E5%85%88%E7%BA%A7%E9%98%9F%E5%88%97" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)。

m) Type Erasure: 实现运行时的多态。

**图像处理库**

a) GIL库：通用图像库

**输入输出库**

a) Assign库：用简洁的语法实现对STL容器赋值或者初始化

b) Format库：实现类似printf的格式化对象，可以把参数格式化到一个字符串，而且是完全类型安全的

c) IO State Savers库：用来保存流的当前状态，自动恢复流的状态等

d) IOStreams库：扩展C++标准库流处理，建立一个流处理框架

e) Program Options库：提供强大的[命令行参数](https://baike.baidu.com/item/%E5%91%BD%E4%BB%A4%E8%A1%8C%E5%8F%82%E6%95%B0" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)处理功能

f) Serialization库：实现C++数据结构的持久化

**跨语言混合编程库**

a) Python库：用于实现Python和C++对象的无缝接口和混合编程

**内存管理库**

a) Pool库：基于简单分隔存储思想实现了一个快速、紧凑的内存池库

b) Smart Ptr库：[智能指针](https://baike.baidu.com/item/%E6%99%BA%E8%83%BD%E6%8C%87%E9%92%88" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)

c) Utility库：小工具的集合

**解析库**

a) Spirit库：基于EBNF范式的LL解析器框架

**编程接口库**

a) Function库：实现一个通用的回调机制，已被收入TR1

b) Parameter库：提供使用参数名来指定函数参数的机制

**综合类库**

a) Compressed Pair库：优化的对pair对象的存储

b) CRC库：实现了[循环冗余校验码](https://baike.baidu.com/item/%E5%BE%AA%E7%8E%AF%E5%86%97%E4%BD%99%E6%A0%A1%E9%AA%8C%E7%A0%81" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)功能

c) Date Time 库：一个非常全面灵活的日期时间库

d) Exception库：针对标准库中异常类的缺陷进行强化，提供<<操作符重载，可以向异常传入任意数据

e) Filesystem库：可移植的文件系统操作库，可以跨平台操作目录、文件，已被TR2接受

f) [Flyweight](https://baike.baidu.com/item/Flyweight" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank) 库：实现享元模式，享元对象不可修改，只能赋值

g) Lexical Cast 库：用于字符串、整数、浮点数的字面转换

h) Meta State Machine库：用于表示UML2有限状态机的库

i) Numeric Conversion 库：用于安全数字转换的一组函数

j) Optional 库：使用容器的语义，包装了可能产生无效值的对象，实现了未初始化的概念

k) Polygon 库：处理平面多边形的一些算法

l) Program Options库：提供强大的命令行参数处理功能

m) Scope Exit库：使用preprocessor库的预处理技术实现在退出[作用域](https://baike.baidu.com/item/%E4%BD%9C%E7%94%A8%E5%9F%9F" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)时资源自动释放

n) Statechart库：提供有限自动[状态机](https://baike.baidu.com/item/%E7%8A%B6%E6%80%81%E6%9C%BA" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)框架

o) Swap库：为交换两个变量的值提供便捷方法

p) System库：使用轻量级的对象封装操作系统底层的错误代码和错误信息，已被TR2接受

q) Timer库：提供简易的度量时间和进度显示功能，可以用于[性能测试](https://baike.baidu.com/item/%E6%80%A7%E8%83%BD%E6%B5%8B%E8%AF%95" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)等需要计时的任务

r) Tribool库：三态布尔逻辑值，在true和false之外引入indeterminate不确定状态

s) Typeof库：模拟C++0x新增加的typeof和auto关键字，以减轻变量类型声明的工作，简化代码

t) Units库：实现了物理学的量纲处理

u) Utility库：小工具集合

v) Value Initialized库：用于保证变量在声明时被正确初始化

w) Chrono库：实现了C++ 0x标准中N2661号建议 [2]  所支持的时间功能。

x) Log库：实现日志功能。

y) Predef库：提供一批统一兼容探测其他宏的预定义宏。 [5]

**编译器问题的变通方案库**

a) Compatibility库：为不符合标准库要求的环境提供帮助

b) Config库：将程序的编译配置分解为三个部分：平台、[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8" \t "https://baike.baidu.com/item/Boost%E5%BA%93/_blank)和标准库，帮助库开发者解决特定平台特定编译器的兼容问题