[Linux编译和安装boost库](http://blog.csdn.net/this_capslock/article/details/47170313)

标签： [linux](http://www.csdn.net/tag/linux)[boost](http://www.csdn.net/tag/boost)[编译器](http://www.csdn.net/tag/%e7%bc%96%e8%af%91%e5%99%a8)

2015-07-31 15:00 2845人阅读 [评论](http://blog.csdn.net/this_capslock/article/details/47170313#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/this_capslock/article/details/47170313#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

C/C++（52） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg Linux（35） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg 开发环境（9） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录[(?)[+]](http://blog.csdn.net/this_capslock/article/details/47170313)

1. 下载boost安装包并解压缩

到<http://www.boost.org/>下载boost的安装包，以boost\_1\_58\_0.tar.gz为例   
下载完成后进行解压缩：

tar zxvf boost\_1\_58\_0.tar.gz

* 1
* 1

2.设置编译器和所选库

先进入解压缩后的目录：

cd boost\_1\_58\_0

* 1
* 1

然后运行bootstrap.sh脚本并设置相关参数：

./bootstrap.sh --with-libraries=all --with-toolset=gcc

* 1
* 1

--with-libraries指定编译哪些boost库，all的话就是全部编译，只想编译部分库的话就把库的名称写上，之间用 , 号分隔即可，可指定的库有以下几种：

| **库名** | **说明** |
| --- | --- |
| atomic |  |
| chrono |  |
| context |  |
| coroutine |  |
| date\_time |  |
| exception |  |
| filesystem |  |
| graph | 图组件和算法 |
| graph\_parallel |  |
| iostreams |  |
| locale |  |
| log |  |
| math |  |
| mpi | 用模板实现的元编程框架 |
| program\_options |  |
| python | 把C++类和函数映射到Python之中 |
| random |  |
| regex | 正则表达式库 |
| serialization |  |
| signals |  |
| system |  |
| test |  |
| thread | 可移植的C++多线程库 |
| timer |  |
| wave |  |

--with-toolset指定编译时使用哪种编译器，[**Linux**](http://lib.csdn.net/base/linux)下使用gcc即可，如果系统中安装了多个版本的gcc，在这里可以指定gcc的版本，比如--with-toolset=gcc-4.4

命令执行完成后看到显示如下即为成功：

Building Boost.Build engine with toolset gcc... tools/build/src/engine/bin.linuxx86\_64/b2

Detecting Python version... 2.6

Detecting Python root... /usr

Unicode/ICU support for Boost.Regex?... not found.

Generating Boost.Build configuration in project-config.jam...

Bootstrapping is done. To build, run:

./b2

To adjust configuration, edit 'project-config.jam'.

Further information:

- Command line help:

./b2 --help

- Getting started guide:

http://www.boost.org/more/getting\_started/unix-variants.html

- Boost.Build documentation:

http://www.boost.org/build/doc/html/index.html

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21

3.编译boost

执行以下命令开始进行boost的编译：

./b2 toolset=gcc

* 1
* 1

编译的时间大概要10多分钟，耐心等待，结束后会有以下提示：

...failed updating 60 targets...

...skipped 21 targets...

...updated 663 targets...

* 1
* 2
* 3
* 1
* 2
* 3

4.安装boost

最后执行以下命令开始安装boost：

./b2 install --prefix=/usr

* 1
* 1

--prefix=/usr用来指定boost的安装目录，不加此参数的话默认的头文件在/usr/local/include/boost目录下，库文件在/usr/local/lib/目录下。这里把安装目录指定为--prefix=/usr则boost会直接安装到系统头文件目录和库文件目录下，可以省略配置环境变量。

安装完毕后会有以下提示：

...failed updating 60 targets...

...skipped 21 targets...

...updated 11593 targets...

* 1
* 2
* 3
* 1
* 2
* 3

最后需要注意，如果安装后想马上使用boost库进行编译，还需要执行一下这个命令：

ldconfig

* 1
* 1

更新一下系统的动态链接库

5.boost使用测试

以boost\_thread为例，[**测试**](http://lib.csdn.net/base/softwaretest)刚安装完的boost库是否能正确使用，测试代码如下：

#include <boost/thread/thread.hpp> //包含boost头文件

#include <iostream>

#include <cstdlib>

using namespace std;

volatile bool isRuning = true;

void func1()

{

static int cnt1 = 0;

while(isRuning)

{

cout << "func1:" << cnt1++ << endl;

sleep(1);

}

}

void func2()

{

static int cnt2 = 0;

while(isRuning)

{

cout << "\tfunc2:" << cnt2++ << endl;

sleep(2);

}

}

int main()

{

boost::thread thread1(&func1);

boost::thread thread2(&func2);

system("read");

isRuning = false;

thread2.join();

thread1.join();

cout << "exit" << endl;

return 0;

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40

在编译程序时，需要加入对boost\_thread库的引用：

g++ main.cpp -g -o main -lboost\_thread

* 1
* 1

如果boost库的安装位置不是在系统目录下，则还需要在编译时加上-I和-L指定boost头文件和库文件的位置

编译成功后运行程序，利用boost实现的多线程任务正确运行：

func1: func2:00

func1:1

func2:1

func1:2

func1:3

func2:2

func1:4

func1:5

func2:3

func1:6

func1:7

func2:4

func1:8

func1:9

func2:5

func1:10

func1:11

func2:6

func1:12

func1:13

func2:7

func1:14

func1:15

func2:8

func1:16

exit