**使用BLADE构建c++工程管理**

字数764 阅读1468 评论1 喜欢3

**一. c++工程依赖管理**

之前在百度一直使用comake2构建c++项目，十分方便。免去了手写Makefile的痛苦，很多项目都不需要从零开始，其中的一个配置类似如下：

WORKROOT('../../../')  
CopyUsingHardLink(True)  
CPPFLAGS('-D\_GNU\_SOURCE -D\_\_STDC\_LIMIT\_MACROS -DVERSION=\\"1.9.8.7\\"')  
CFLAGS('-g -pipe -W -Wall -fPIC')  
CXXFLAGS('-g -pipe -W -Wall -fPIC')  
INCPATHS('. ./include ./output ./output/include')  
CONFIGS('lib2-64/dict@dict\_3-1-15-0\_PD\_BL')  
CONFIGS('lib2-64/ullib@ullib\_3-1-41-0\_PD\_BL')  
CONFIGS('public/configure@configure\_1-2-1-0\_PD\_BL')  
CONFIGS('public/ependingpool@ependingpool\_1-0-6-0\_PD\_BL')  
Application('test',Sources(user\_sources))  
StaticLibrary('test',Sources(user\_sources),HeaderFiles(user\_headers))  
SharedLibrary('test',Sources(user\_sources),HeaderFiles(user\_headers))

基本思路是：新的项目相当于代码树的叶子节点，它可以依赖现有代码树上有权限的任意的一个字节点代码。comake2生成Makefile文件进行编译。

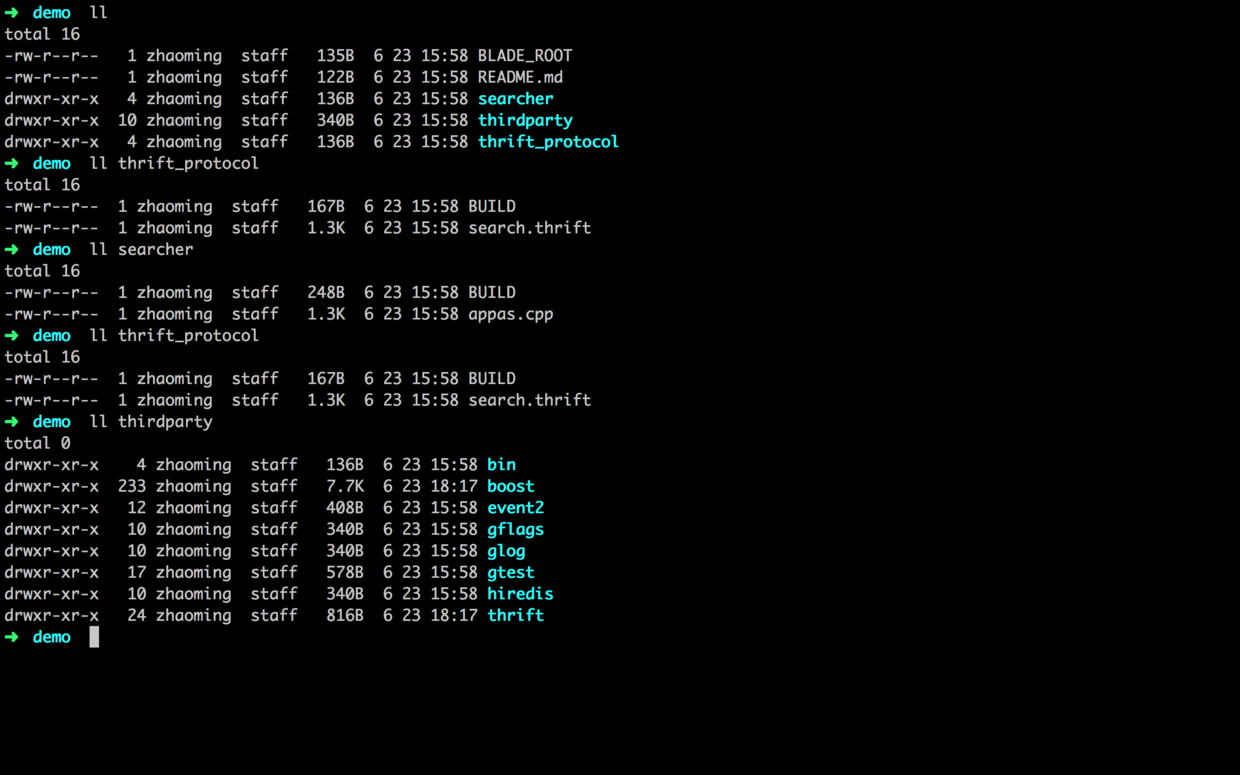
**WORKROOT**： 指定了这颗代码树的根节点的相对位置  
**CONFIGS**：指定了你要依赖的库的位置，以及具体的版本，如果不指定则依赖trunk的代码，这个和maven的snapshot版本等概念类似。  
**Application**: 输出二进制可运行  
**StaticLibrary**：输出静态库  
**SharedLibrary**：输出动态库

很可惜comake2一直没有开源。  
类似的工具在其他的语言已经有了很好的解决方案  
nodejs － npm  
php － composer  
python － pip  
java － maven

**二. BLADE － from 腾讯**

[blade@github](https://github.com/chen3feng/typhoon-blade)  
[一个结合第三方lib和thrift的示例](http://pan.baidu.com/s/1kTte6bl)

* 原理  
  不同于COMAKE2使用make, BLADE使用scons进行编译  
  makefile和scons的区别在这里有着详细的说明：[make vs scons](http://pan.baidu.com/s/1jGowP5G" \t "_blank)
* 我们在实际项目中使用了blade，总结下优点如下：
  1. 配置简单，只需要配置向下一层的依赖关系，因此可以逐层构建c++代码仓库
  2. 支持thrift和proto自动编译连接的功能
* 下图说明一个binary代码组织结构，注意：BLADE\_ROOT文件，作为工程root目录的标记



*总体代码目录组织.png*

* 二进制依赖配置



*binary.png*

其中  
**name** : blade作为依赖的最小单元，也是编译的二进制的目标名称  
**src** : cpp文件列表  
**deps**:依赖全局的pthread.a，依赖root目录/thrift\_protocol目录  
**optimize**: 编译的flag设置，是否使用c++0x

* 使用小技巧
  1. 如果编译失败，开启--verbose模式，会有详细编译连接命令。很多时候是由于BLADE\_ROOT没有合理配置
  2. 如果链接失败，可以开启ld\_debug=all, 定位  
     Valid options for the LD\_DEBUG environment variable are:

**libs** display library search paths  
**reloc** display relocation processing  
**files** display progress for input file  
**symbols** display symbol table processing  
**bindings** display information about symbol binding  
**versions** display version dependencies  
**scopes** display scope information  
**all** all previous options combined  
**statistics** display relocation statistics  
**unused** determined unused DSOs  
**help** display this help message and exit

**三. 好消息－google开源了多语言，多平台构建的工具－Bazel**

[Bazel@github](https://github.com/google/bazel)  
[Bazel主页](http://bazel.io/)

* 该项目还处在alpha阶段，不过据说google内部的服务器和移动端的工程都使用该工具管理，十分值得期待。后面有时间补充下这块的内容和demo