目录

[一、Windows 和linux下执行单文件 1](#_Toc532405260)

[二、Windows 和linux下执行多文件or项目 2](#_Toc532405261)

[三、解决多文件编译的困难：makefile 3](#_Toc532405262)

[四、Cmake工具:编译运行文件 4](#_Toc532405263)

[五、使用cmake方便的编译执行单文件Demo 5](#_Toc532405264)

[六、使用cmake方便的编译执行多文件项目 6](#_Toc532405265)

[七、一个复杂的例子：关于CMakelists子目录 and 生成库 7](#_Toc532405266)

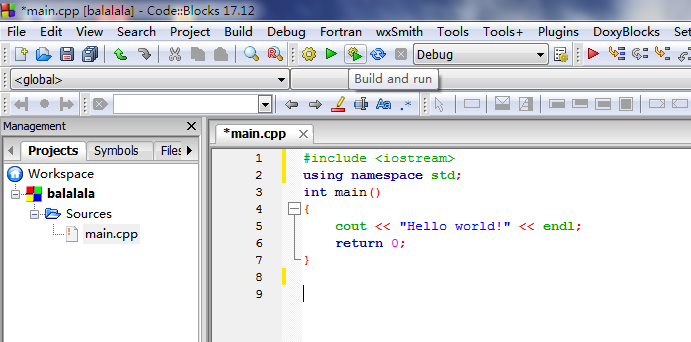
[八、总结cmakelists.txt的整理内容 8](#_Toc532405267)

### 一、Windows 和linux下执行单文件

在windows环境下，大家都熟悉怎么编写并执行一份代码：

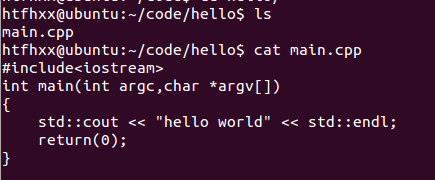
1. 先打开编译器例如codeblocks编写源代码，例如一个c++文件；
2. 点击编译按钮，编译代码，生成.o的目标文件。
3. 点击执行按钮，生成.exe的可执行文件，运行完毕。

例如codeblocks下的build & run：

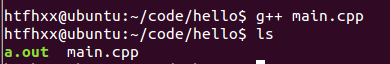


在linux环境下呢？

用vim编辑器编写代码，得到一个文本文件 main.cpp



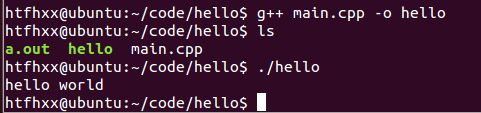
1. 使用g++编译main.cpp得到a.out文件



1. 执行a.out文件，执行完毕得到结果



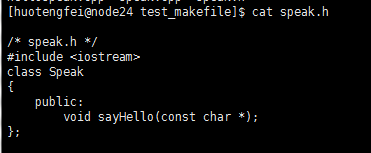
当然，更普遍的是使用-o来编译和执行的

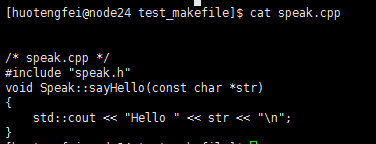


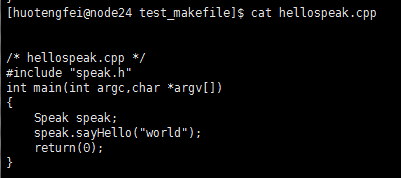
### 二、Windows 和linux下执行多文件or项目

Windows自不必多说，在编译器下编译运行main函数即可

至于linux，有如下三个文件speak.h speak.cpp hellospeak.cpp







编译执行多个文件：

g++ hellospeak.cpp speak.cpp -o hellospeak



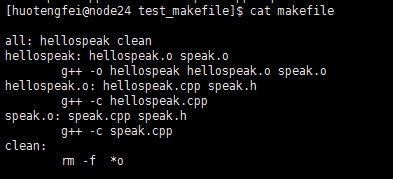
这个时候会发现，如果源文件太多，一个一个编译时就会特别麻烦。于是人们想到了制作一种类似批处理的程序，来批处理编译源文件，于是就有了make工具。它是一个自动化的编译工具，你可以使用一条命令实现完全编译。但是你需要编写一个规则文件，make依据它来批处理编译，这个文件就是makefile。

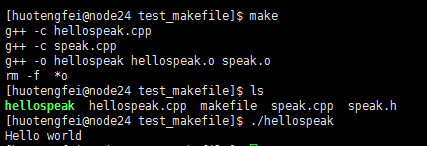
### 三、解决多文件编译的困难：makefile

文件下包含一个头文件和两个cpp文件，以及一个写好的makefile：



Makefile大致内容就是要编译两个文件得到hellospeak.o和speak.o，再生成可执行文件hellospeak，最后删掉.o文件。

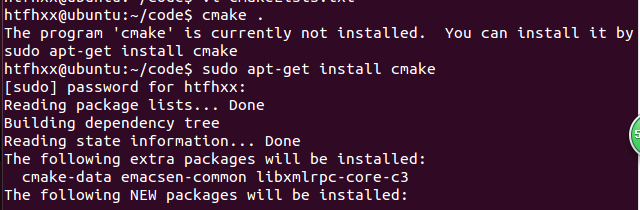




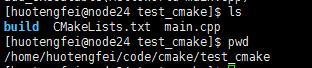
对于一个大工程，编写makefile实在是件复杂的事，于是就出现了cmake工具，它能够输出各种各样的makefile或者project文件,从而帮助程序员减轻负担。但是随之而来也就是编写cmakelist文件，它是cmake所依据的规则。

### 四、Cmake工具:编译运行文件

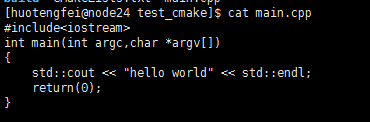
如果没有安装的话就通过sudo apt-get install cmake命令安装cmake工具：



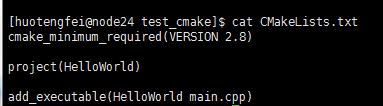
准备好cmakelist.txt文件和要执行的main.cpp，以及一个build文件，用于放入cmake编译的繁多的中间文件：



要执行的main.cpp：

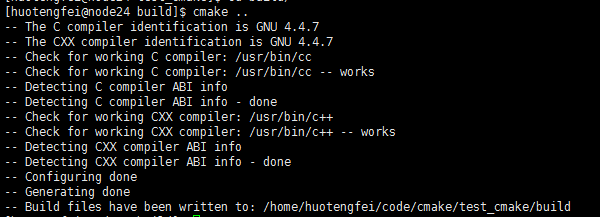


cmakelist.txt文件（内容撰写待会再说）：

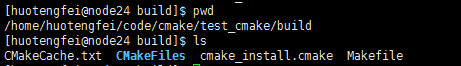


第一步，cmake + (cmakelists.txt所在文件夹)。

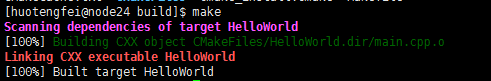
此处“..”指的是上一级文件夹，会从文件夹中找到cmakelists.txt：



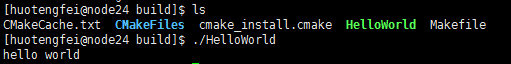
系统自动生成了：CMakeFiles, CMakeCache.txt, cmake\_install.cmake 等文件，并且生成了Makefile



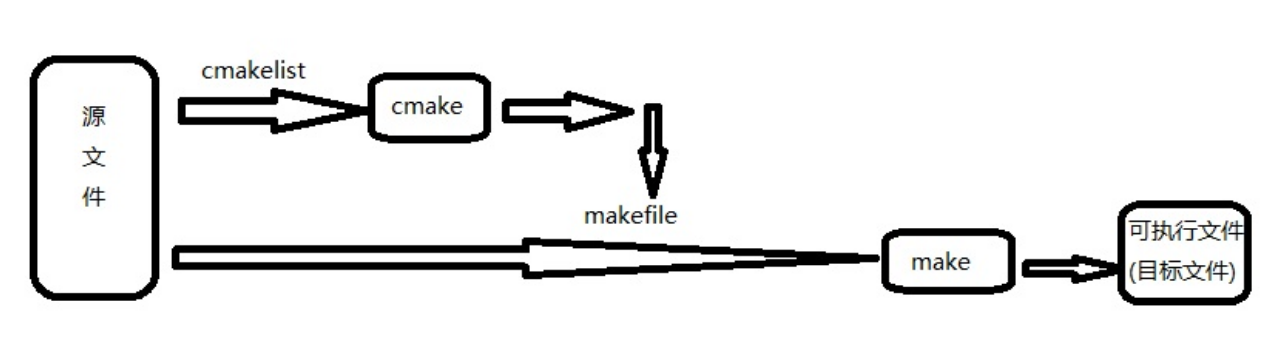
进行工程的实际构建，在这个目录输入make 命令，大概会得到如下的彩色输出：



到这里就已经编译完成了，接下来执行这个项目得到hello world的输出：

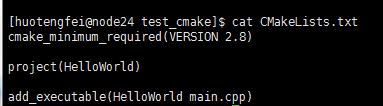


即，整个流程为（网图，侵删）：



### 五、使用cmake方便的编译执行单文件Demo

先分析上一例子中的cmakelists.txt：



cmake\_minimum\_required(VERSION 2.8)

//指的是支持的cmake版本，可省略，但是为了方便后人，尽量加上自己所用的版本。

project(HelloWorld)

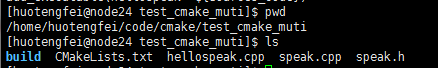
//指定项目名称，编译完成后生成的名字就是HelloWorld

add\_executable(HelloWorld main.cpp)

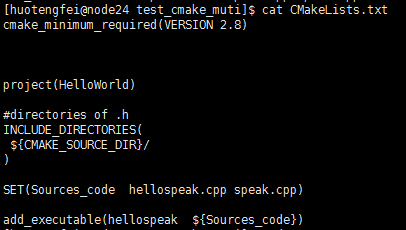
//加入执行文件，此处是单文件，待会展开来讲

### 六、使用cmake方便的编译执行多文件项目

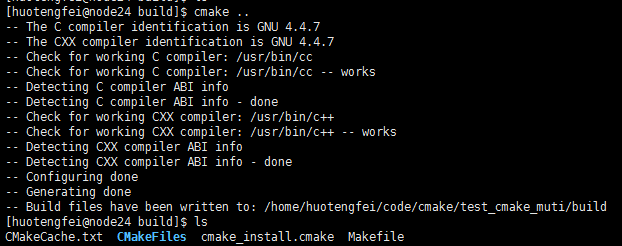
两个cpp文件一个.h文件和一个build，这次我们试着用cmake编译执行多文件。

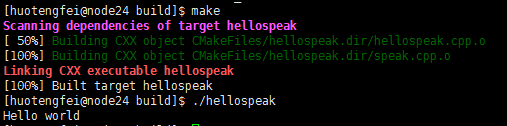


这个example的cmakelists.txt里包含了之前讲的版本和项目名，加了一个include\_directories，这个参数是把.h文件所在目录包含进去，可以是一堆的.h文件。其中cmake\_source\_dir是系统变量，可以通过set关键字来设置，默认来说是cmakelists.txt所在的文件。

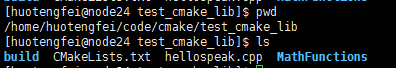


接着就是执行两个cpp文件，当然两个cpp文件也可以通过set来设置成变量Sources\_code.





### 七、一个复杂的例子：关于CMakelists子目录 and 生成库

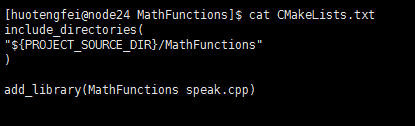


这个文件夹下包含build文件、一个要执行的cpp文件和CMakelists.txt。除此之外，还有一个MathFunctions文件夹。

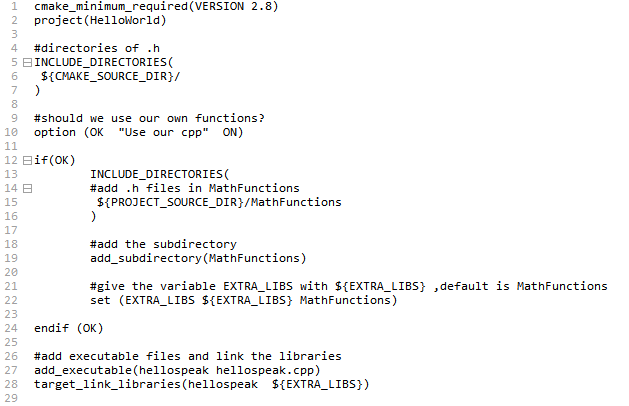
先来看子目录MathFunctions：



这里的CMakeLists.txt只有几行，将speak.cpp中的函数生成一个MathFunctions库。

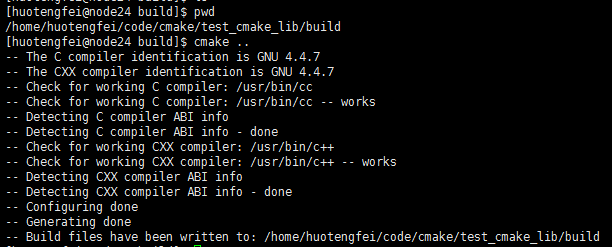


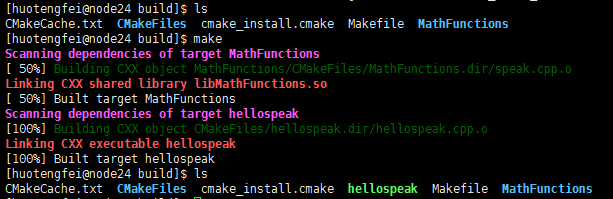
在顶层目录下的CMakeLists.txt中，通过add\_subdirectories加入子目录的CMakeLists.txt。

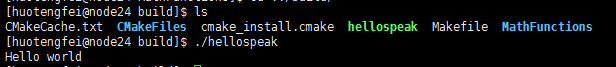


前两行是版本和项目名，五六行加入.h文件的目录，10行设置一个变量默认为ON,12行判断是否OK，如果OK为ON的话就可以执行13-23行。

13-23行，是确定使用自己本地（即MathFunctions文件夹中的库），分别是加入.h文件、加入子目录（划重点，下面讲）、设置EXTRA\_LIBS变量，如果未设置28行就不再链接这个库。







### 八、总结cmakelists.txt的整理内容

cmakelists.txt内容不需区分大小写。

一般的cmakelists.txt的编写，包括以下几部分：

1. 指定cmake版本，就像上面所说的：为了方便后人，尽量加上自己所用的版本cmake\_minimum\_required(VERSION 2.8)
2. 指定项目的名称，一般和项目的文件夹名称对应

project(HelloWorld)

1. 设置环境变量 **SET(变量名 变量值)**

一般包括（但不仅仅包括）：

**CMAKE\_C\_COMPILER**：指定C编译器

**CMAKE\_CXX\_COMPILER**：指定C++编译器

**CMAKE\_C\_FLAGS**：编译C文件时的选项，如-g；也可以通过add\_definitions添加编译选项

**EXECUTABLE\_OUTPUT\_PATH**：可执行文件的存放路径

**LIBRARY\_OUTPUT\_PATH**：库文件路径

**CMAKE\_BUILD\_TYPE**:：build 的类型(Debug, Release, ...)

变量很多很复杂，根据需要使用即可，可以从官方文档中查找：<https://cmake.org/cmake/help/v3.0/manual/cmake-variables.7.html>

当然还有一些自己定义的变量名，也用set设置

1. LINK\_DIRECTORIES 添加需要链接的库文件目录,即链接库搜索路径

link\_directories(directory1 directory2 ...)

1. 添加可执行文件要链接的库文件的名称

TARGET\_LINK\_LIBRARIES(PROJECT\_NAME libname.so)

1. 头文件目录

INCLUDE\_DIRECTORIES(

Include

)

如果文件夹较多，则可以这样写：

INCLUDE\_DIRECTORIES(

${CMAKE\_SOURCE\_DIR}/include/

${CMAKE\_SOURCE\_DIR}/include/a/

${CMAKE\_SOURCE\_DIR}/include/b/

)

1. 源文件目录

AUX\_SOURCE\_DIRECTORY(src DIR\_SRCS)

1. 添加要编译的可执行文件

ADD\_EXECUTABLE(PROJECT\_NAME TEST\_CPP)

1. **生成动态库or 静态库**

这里多说两句，用cmake生成静态动态库，是将在cmakelists.txt文件中加入的源文件头文件等等，生成一个类似于.h/.a的文件

这与<8>中加入想要编译的可执行文件是二选一的关系。

add\_library(person SHARED ${srcs})

add\_library(person\_static STATIC ${srcs})

install（TARGETS）

创建规则以将列出的目标安装到给定目录中。