**【Cmake函数详解---CmakeList.txt---Makefile】**

**关系：CmakeList.txt是配置cmake的文件，cmake为构建工具，按照配置于运行可以产生Makefile**

<https://blog.csdn.net/whahu1989/article/details/82078563?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522162364243316780261997480%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=162364243316780261997480&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-82078563.first_rank_v2_pc_rank_v29&utm_term=camke&spm=1018.2226.3001.4187>

# 【函数】

**1、cmake\_minimum\_required(VERSION 2.8) #要求最低版本2.8（cmake –version可查看版本）**

**2、project(demo) #工程名为demo**

**在每个写了project指令的Cmakelists.txt文件里，会生成PROJECT\_SOURCE\_DIR和PROJECT\_BINARY\_DIR**

**PROJECT\_BINARY\_DIR: 编译工程的文件夹路径（build）**

**PROJECT\_SOURCE\_DIR：最后一个有project语句的CmakeLists.txt 所在的文件夹**

**可用输出信息message输出路径信息**

**3、add\_executable(xx xx1.c xx2.c … ) #将xx1.c、xx2.c源文件一起编译为xx**

**4、aux\_source\_directory(dir var) #dir：存放的源文件的目录，var：SRC\_LIST，将dir目录的源文件all存放在var变量里，${ SRC\_LIST }：调用SRC\_LIST变量**

**5、set(SRC\_LIST**

**./main.c**

**./test1.c**

**) #将当前目录下的main.c、test1.c存放在于变量SRC\_LIST**

**set(SRC\_LIST ${PROJECT\_SOURCE\_DIR}/test/test1.c) #将test1.c 放在SRC\_LIST**

**set(EXECUTABLE\_OUTPUT\_PATH ${PROJECT\_SOURCE\_DIR}/bin) # EXECUTABLE\_OUTPUT\_PATH：编译后的可执行文件存放路径**

**6、不同目录下存在源文件解决方案：多级CmakeList.txt**

**include\_directories(xx xx …) #指定各个存放头文件文件夹的路径（不包含该CmakeList.txt所在的文件夹）**

**link\_directories(xx xx …) #添加库文件搜索路径**

**7、add\_subdirectory(x1 x2) #x1指定源文件子目录，x2生成文件的目录（可省）（在外层CmakeList.txt使用）**

**9、编译成动态库（.so）/静态库（.a）：**

**编译库不需要声明该.c文件的.h文件，若该.c文件依赖其他.h则需要包含。**

**链接静态库生成的可执行文件，可执行文件编译结束后，不依赖任何包，将库删除，复制到任意位置都可以运行；而动态库不行，可执行文件编译结束后，删除依赖库，就不能运行了！！！**

**①add\_library(x1 SHARED /STATIC ${SRC\_LIST}) #第一个指定库的名字，第二个指定动态还是静态，第三个指定源文件，x1不用加lib开头：x1设置为test，则编译后的库为libtest.so/libtest.a**

**②set\_target\_properties(x2 PROPERTIES OUTPUT\_NAME ”x3”) #可以将静态库/动态库名字设置为一样的x3，只是后缀不同，但x1与x2必须一样**

****

**③set(LIBRARY\_OUT\_PATH ${PROJECT\_SOURCE\_DIR}/lib)**

**10、链接**

**①find\_library(x1 x2 ${ PROJECT\_SOURCE\_DIR }/….) #查找某路径下的库x2，并将路径存在x1中**

**（x2可以不带前lib和后.so或.a; 若带后缀，必须带前缀）**

**注意：莫名其妙的毛病时，删除CmakeCache.txt文件，重新构建！！！**

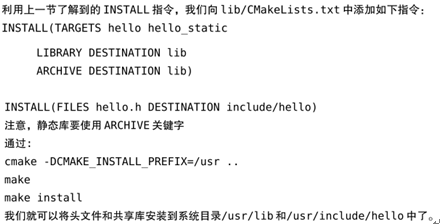
**②target\_link\_libraries(xx ${x1}) #将x1（含有库路径的x1）与xx进行链接**

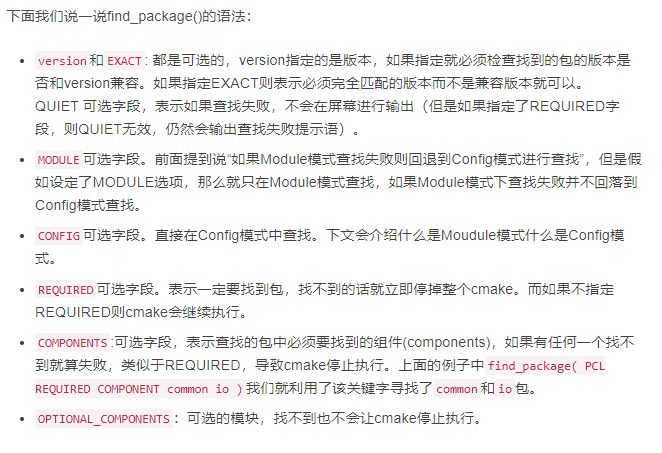
**11、add\_compile\_options(-std=c++11 -Wall -o2 …) #添加编译选项**

**12、option(xx “…” NO/OFF) #添加控制选项（选择性编译文件），根据xx的值（NO/OFF），用if语句判断是否编译， 通过cmake .. –Dxx=NO && make可以重新设置xx值并编译**

**13、INSTALL() CMAKE\_INSTALL\_PREFIX #用于指定安装规则**

**install指定的目录，只有使用make install编译才会生效**

****

****

# 【语法】

1、函数名大小写无关，参数大小写有关；

2、调用变量值：${xx}，但在if语句中，只能if(xx)；

3、参数用空格或分号分割；

4、编译c文件必须.c结尾，编译c++文件必须以.cpp结尾