

## cmake 添加头文件目录，链接动态、静态库

罗列一下cmake常用的命令。

CMake支持大写、小写、混合大小写的命令。

### 1. 添加头文件目录**INCLUDE\_DIRECTORIES**

语法：

```
include_directories([AFTER|BEFORE] [SYSTEM] dir1 [dir2 ...])
```

它相当于g++选项中的-I参数的作用，也相当于环境变量中增加路径到CPLUS\_INCLUDE\_PATH变量的作用。

```
include_directories ../../../../thirdparty/comm/include)
```

### 2. 添加需要链接的库文件目录**LINK\_DIRECTORIES**

语法：

```
link_directories(directory1 directory2 ...)
```

它相当于g++命令的-L选项的作用，也相当于环境变量中增加LD\_LIBRARY\_PATH的路径的作用。

```
link_directories("/home/server/third/lib")
```

### 3. 查找库所在目录**FIND\_LIBRARY**

语法：



A **short**-hand signature is:

```
find_library (<VAR> name1 [path1 path2 ...])
```

The general signature is:

```
find_library (
    <VAR>
    name | NAMES name1 [name2 ...] [NAMES_PER_DIR]
    [HINTS path1 [path2 ... ENV var]]
    [PATHS path1 [path2 ... ENV var]]
    [PATH_SUFFIXES suffix1 [suffix2 ...]]
    [DOC "cache documentation string"]
    [NO_DEFAULT_PATH]
    [NO_CMAKE_ENVIRONMENT_PATH]
    [NO_CMAKE_PATH]
    [NO_SYSTEM_ENVIRONMENT_PATH]
    [NO_CMAKE_SYSTEM_PATH]
    [CMAKE_FIND_ROOT_PATH_BOTH |
     ONLY_CMAKE_FIND_ROOT_PATH |
     NO_CMAKE_FIND_ROOT_PATH]
)
```



例子如下:

```
FIND_LIBRARY(RUNTIME_LIB rt /usr/lib /usr/local/lib NO_DEFAULT_PATH)
```

cmake会在目录中查找，如果所有目录中都没有，值RUNTIME\_LIB就会被赋为NO\_DEFAULT\_PATH

#### 4. 添加需要链接的库文件路径**LINK\_LIBRARIES**

语法:

```
link_libraries(library1 <debug | optimized> library2 ...)
```



# 直接是全路径

```
link_libraries("/home/server/third/lib/libcommon.a")
```

# 下面的例子，只有库名，cmake会自动去所包含的目录搜索

```
link_libraries(iconv)
```

# 传入变量

```
link_libraries(${RUNTIME_LIB})
```

# 也可以链接多个

```
link_libraries("/opt/MATLAB/R2012a/bin/glnxa64/libeng.so" "/opt/MATLAB/R2012a/bin/glnxa64/libmx.so")
```



可以链接一个，也可以多个，中间使用空格分隔。

#### 5. 设置要链接的库文件的名称**TARGET\_LINK\_LIBRARIES**

语法:

```
target_link_libraries(<target> [item1 [item2 [...]]]
                      [[debug|optimized|general] <item>] ...)
```



# 以下写法都可以:

```
target_link_libraries(myProject comm) # 连接libhello.so库，默认优先链接动态库
```

```
target_link_libraries(myProject libcomm.a) # 显示指定链接静态库
```

```
target_link_libraries(myProject libcomm.so) # 显示指定链接动态库
```

# 再如:

```
target_link_libraries(myProject libcomm.so) # 这些库名写法都可以。
```

```
target_link_libraries(myProject comm)
```

```
target_link_libraries(myProject -lcomm)
```



#### 6. 为工程生成目标文件


语法:

```
add_executable(<name> [WIN32] [MACOSX_BUNDLE]
               [EXCLUDE_FROM_ALL]
               source1 [source2 ...])
```

简单的例子如下:

```
add_executable(demo
               main.cpp
               )
```

## 6. 最后贴一个完整的例子



```
cmake_minimum_required (VERSION 2.6)

INCLUDE_DIRECTORIES(../../thirdparty/comm)

FIND_LIBRARY(COMM_LIB comm ../../thirdparty/comm/lib NO_DEFAULT_PATH)
FIND_LIBRARY(RUNTIME_LIB rt /usr/lib /usr/local/lib NO_DEFAULT_PATH)

link_libraries(${COMM_LIB} ${RUNTIME_LIB})

ADD_DEFINITIONS(
-O3 -g -W -Wall
-Wunused-variable -Wunused-parameter -Wunused-function -Wunused
-Wno-deprecated -Woverloaded-virtual -Wwrite-strings
-D__WUR= -D_REENTRANT -D_FILE_OFFSET_BITS=64 -DTIXML_USE_STL
)

add_library(lib_demo
            cmd.cpp
            global.cpp
            md5.cpp
            )

link_libraries(lib_demo)
add_executable(demo
               main.cpp
               )

# link library in static mode
target_link_libraries(demo libuuid.a)
```



另外, 使用cmake生成makefile之后, make edit\_cache可以编辑编译选项。