cmake 学习笔记(二) - Mingz技术博客 - 博客频道

分类：

c/c++ (*138*)

linux/unix (*261*)

http://blog.csdn.NET/dbzhang800/article/details/6329068

在 Cmake学习笔记一 中通过一串小例子简单学习了cmake 的使用方式。

这次应该简单看看语法和常用的命令了。

简单的语法

- 注释

```
# 我是注释
```

- 命令语法

```
COMMAND(参数1 参数2 ...)
```

- 字符串列表

```
A;B;C # 分号分割或空格分隔的值
```

- 变量(字符串或字符串列表)

set(Foo a b c)	设置变量 Foo
command(${Foo})	等价于 command(a b c)
command("${Foo}")	等价于 command("a b c")
command("/${Foo}")	转义，和 a b c无关联

- 流控制结构

```
IF()...ELSE()/ELSEIF()...ENDIF()
```

WHILE()...ENDWHILE()
FOREACH()...ENDFOREACH()

- 正则表达式

部分常用命令

INCLUDE_DIRECTORIES("dir1" "dir2" ...)	头文件路径，相当于编译器参数 -Idir1 -Idir2
LINK_DIRECTORIES("dir1" "dir2")	库文件路径。注意： 由于历史原因，相对路径会原样传递给链接器。 尽量使用FIND_LIBRARY而避免使用这个。
AUX_SOURCE_DIRECTORY(“sourcedir” variable)	收集目录中的文件名并赋值给变量
ADD_EXECUTABLE	可执行程序目标
ADD_LIBRARY	库目标
ADD_CUSTOM_TARGET	自定义目标
ADD_DEPENDENCIES(target1 t2 t3)	目标target1依赖于t2 t3
ADD_DEFINITIONS("-Wall -ansi")	本意是供设置 -D... /D... 等编译预处理需要的宏定义参数，对比 REMOVE_DEFINITIONS()
TARGET_LINK_LIBRARIES(target-name lib1 lib2 ...)	设置单个目标需要链接的库
LINK_LIBRARIES(lib1 lib2 ...)	设置所有目标需要链接的库
SET_TARGET_PROPERTIES(...)	设置目标的属性 OUTPUT_NAME, VERSION, ...
MESSAGE(...)	
INSTALL(FILES “f1” “f2”DESTINATION .)	DESTINATION 相对于 ${CMAKE_INSTALL_PREFIX}
SET(VAR value [CACHE TYPE DOCSTRING [FORCE]])	

LIST(APPEND|INSERT|LENGTH|GET| REMOVE_ITEM|REMOVE_AT|SORT ...)	列表操作
STRING(TOUPPER|TOLOWER|LENGTH| SUBSTRING|REPLACE|REGEX ...)	字符串操作
SEPARATE_ARGUMENTS(VAR)	转换空格分隔的字符串到列表
FILE(WRITE|READ|APPEND|GLOB| GLOB_RECURSE|REMOVE|MAKE_DIRECTORY ...)	文件操作
FIND_FILE	注意 CMAKE_INCLUDE_PATH
FIND_PATH	注意 CMAKE_INCLUDE_PATH
FIND_LIBRARY	注意 CMAKE_LIBRARY_PATH
FIND_PROGRAM	
FIND_PACKAGE	注意 CMAKE_MODULE_PATH
EXEC_PROGRAM(bin [work_dir] ARGS <..> [OUTPUT_VARIABLE var] [RETURN_VALUE var])	执行外部程序
OPTION(OPTION_VAR “description” [initial value])	

变量

这三个变量指代的内容是一致的，是工程顶层目录

- CMAKE_BINARY_DIR
- PROJECT_BINARY_DIR
- <projectname>_BINARY_DIR

这三个变量指代的内容是一致的，如果是in source编译，指得就是工程顶层目录，如果 是out-of-source编译，指的是工程编译发生的目录

- **CMAKE_CURRENT_SOURCE_DIR**

指的是当前处理的CMakeLists.txt所在的路径。

- **CMAKE_CURRRENT_BINARY_DIR**

如果是in-source编译，它跟CMAKE_CURRENT_SOURCE_DIR一致，如果是out-ofsource 编译，他指的是target编译目录。

- **CMAKE_CURRENT_LIST_FILE**

输出调用这个变量的CMakeLists.txt的完整路径

CMAKE_BUILD_TYPE

控制 Debug 和 Release 模式的构建

- CMakeList.txt文件

```
SET(CMAKE_BUILD_TYPE Debug)
```

- 命令行参数

```
cmake DCMAKE_BUILD_TYPE=Release
```

也可以通过指令ADD_DEFINITIONS()添加

CMAKE_INCLUDE_PATH

- 配合 FIND_FILE() 以及 FIND_PATH() 使用。如果头文件没有存放在常规路径(/usr/include, /usr/local/include 等)，

则可以通过这些变量就行弥补。如果不使用 FIND_FILE 和 FIND_PATH的话，CMAKE_INCLUDE_PATH，没有任何作用。

- **CMAKE_LIBRARY_PATH**

配合 FIND_LIBRARY() 使用。否则没有任何作用

- **CMAKE_MODULE_PATH**

cmake 为上百个软件包提供了查找器(finder):FindXXXX.cmake

当使用非cmake自带的finder时，需要指定finder的路径，这就是CMAKE_MODULE_PATH，配合FIND_PACKAGE()使用

CMAKE_INSTALL_PREFIX

控制make install是文件会安装到什么地方。默认定义是/usr/local 或 %PROGRAMFILES%

BUILD_SHARED_LIBS

如果不进行设置，使用ADD_LIBRARY且没有指定库类型，默认编译生成的库是静态库。

UNIX 与 WIN32

- UNIX，在所有的类UNIX平台为TRUE，包括OS X和cygwin
- WIN32，在所有的win32平台为TRUE，包括cygwin