**测试环境：Windows 10（Ubuntu）+MinGw（gcc/g++/Ninja.exe）+ Visual Studio 2019 +Clion（克利翁）**

**插图以及内容来自书籍《Modern CMake For C++》**

**辅助工具：ChatGPT**

**Cmake入门02-CMake语言：**

本节主要的内容：

* Comments和Message注释命令；
* 变量和参数，以及作用域；
* 真假值判断，比较操作符，条件表达式；
* 循环语句；
* 宏与函数；
* Include命令；

掌握了这些命令，基本就掌握了大部分cmake基础语法；

**前提：**

确保电脑已经安装了cmake：

cmake --version

**执行一个cmake脚本：**



**命令的基本规则：**

* 命令名称不区分大小写；
* 语句不需要加分号结束；

**Comments注释命令：**

单行注释和多行注释：

* 单行注释，用#表示；
* 多行注释是，#[=[ #]=] 其中等于号=，可以是0-N个；

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

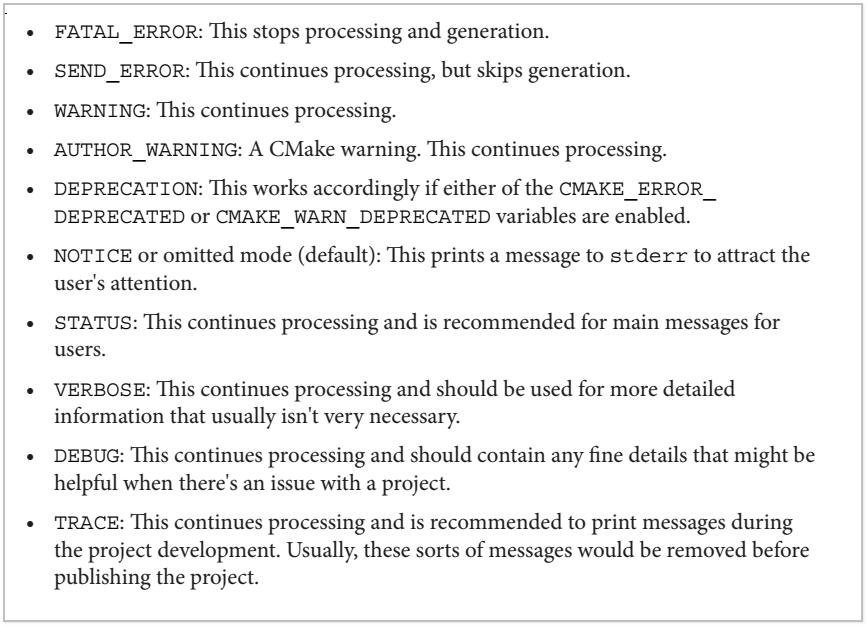
**message命令：**



图示

中度可信度描述已自动生成

**Mode：（需要主要的就是直接结束脚本）**



**Command arguments命令参数：**

图形用户界面, 应用程序

描述已自动生成

1. **Bracket参数：（括号参数）**

图形用户界面, 文本, 应用程序

描述已自动生成

表格

中度可信度描述已自动生成

1. **quoted参数：（带引号参数）**

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

1. **unquoted参数：（不带引号参数）**

图示

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

**关于变量的规定：**

* 变量名称区分大小写，几乎可以包含任何字符。
* 所有变量都在内部存储为字符串，即使某些命令可以将它们解释为其他数据类型（甚至列表！）的值。
* 基本的变量操作命令是set() 和unset()，但还有其他命令可以影响变量，例如string() 和list()。
* 建议变量名称中仅使用字母数字字符、短划线 （-） 和下划线 （\_）

**结论：变量尽量不要用个空格，用小写字母+“\_”或是”-”分隔开来；**

**设置变量和清理变量：**

设置变量：set(my\_var “hello world”)

取消变量：unset(my\_var)

**使用变量：**

${ my\_var }



**1.普通变量：**

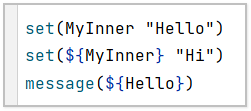
文本

描述已自动生成

注意：

${MyString1}在求值时，CMake将遍历作用域堆栈，并将${MyString1}替换为一个值，

如果没有找到变量，则替换为空字符串



**2.环境变量：**

图形用户界面, 文本, 应用程序

描述已自动生成

**3.缓存变量：**

图形用户界面, 文本, 应用程序

描述已自动生成

其他的变量变种：

**List变量：**

**通过引号定义的变量和非引用定义的变量**

图形用户界面, 文本, 应用程序

描述已自动生成

**List相关的命令：**

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

**List的举例：**

电脑萤幕的截图

描述已自动生成

**根据作用域划分的变量类型：**

图形用户界面, 文本, 应用程序

描述已自动生成

**变量的作用域：**

图形用户界面, 应用程序

描述已自动生成

**函数作用域：**在函数内部创建或修改的任何变量都限制在该函数的作用域内，并且在函数之外不可访问（相当于传值）

**一个例子看变量作用域：**

表格

描述已自动生成

**改变作用域的范围：**

图形用户界面, 文本, 应用程序

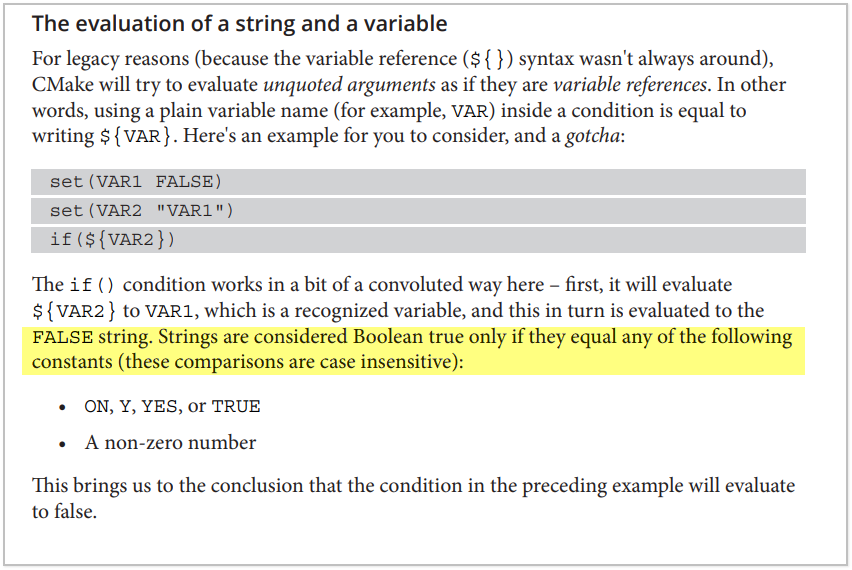
描述已自动生成

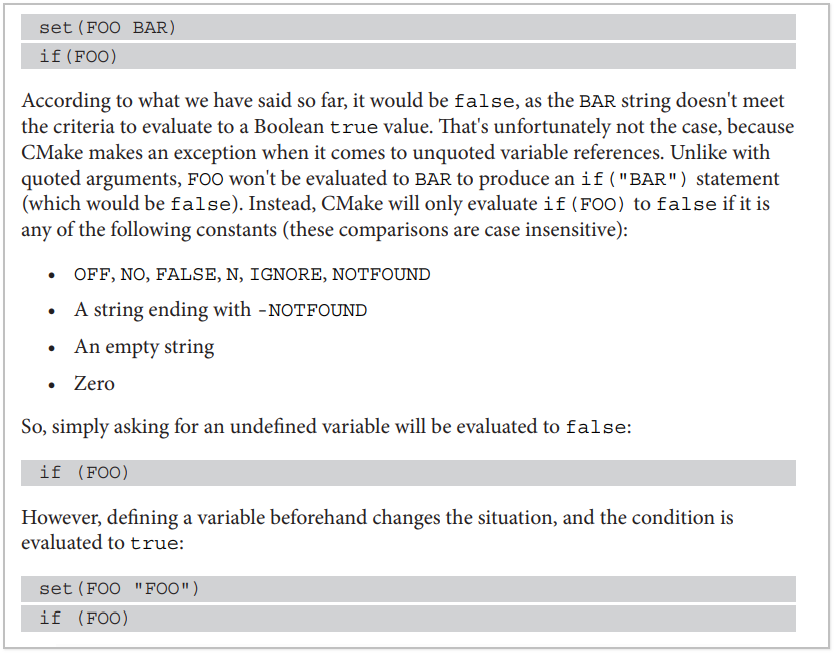
**真假值判断：（不是太好理解）**

图形用户界面, 文本

描述已自动生成

**仅当字符串等于以下任何常量（这些比较不区分大小写）时，字符串才被视为布尔值真：**





**当我们通过一个变量去判断的时候，只有这个变量的值为以上的值时候，这个变量才会flase；**

**如果您认为不带引号的参数的行为令人困惑，**

**请将变量引用包装在带引号的参数中：if ('${FOO}')。**

**这将导致在将提供的参数传递到 if() 命令之前进行参数评估，并且行为将与字符串评估一致。**

**常见陷阱：**

电脑屏幕的手机截图

描述已自动生成

**换句话说，在条件中使用普通变量名称（例如 VAR）等同于编写 ${VAR}**

**判断是否定义了某个变量：**

图形用户界面, 文本, 应用程序

描述已自动生成

**逻辑表达式：**

图形用户界面, 文本, 应用程序

描述已自动生成

**Cmake中的比较操作：**

图示, 文本, 信件

描述已自动生成

* VERSION\_LESS\_EQUAL 版本号比较；
* STREQUAL 字符串比较；

**宏命令以及函数：**

**macros** 相当于c语言的宏命令，不会创建作用域，会将参数转换成字符串或者进行相应的处理，并返回结果，可以不需要用 **return()** 语句返回值，如果需要返回值可以使用参数传递的方式返回

**functions** 只能接受预定义的参数，而无法像 **macros** 一样接受任意数量和类型的参数。**functions** 必须通过 **return()** 来返回一个值，否则将返回空值。

**宏命令：**

图形用户界面

低可信度描述已自动生成

**宏命令举例：**

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

**函数：**

表格

中度可信度描述已自动生成

**函数举例：**

文本, 表格

描述已自动生成

**条件语句：**

图形用户界面, 表格

中度可信度描述已自动生成

**循环语句：**

**While循环：**

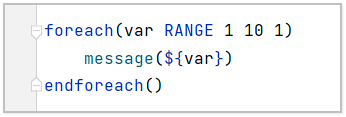
图形用户界面, 文本, 应用程序

描述已自动生成

**Foreach循环：**

文本, 应用程序, 电子邮件

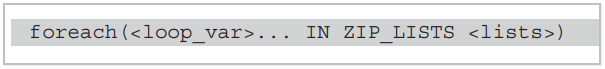
描述已自动生成



图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

**变种：ZIP\_LISTS**



图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

* 执行命令 **continue()** 跳过了这次循环
* 使用 **break()** 命令可以跳出当前循环
* 使用 **return()** 命令可以跳出当前函数或macro

举例说明：

电脑萤幕的截图

描述已自动生成

**Include命令：**

CMake 中的 **include** 命令用于包含其他 CMake 脚本或文件，类似于程序中的 **#include** 命令。



**Include\_guard命令：**

**include\_guard()** 是 CMake 中一个用于生成头文件保护宏的命令，该命令主要用于保证头文件只被编译一次。

在 C/C++ 中，为了防止头文件被重复包含，可以在头文件的开始处加上 **#ifndef** 的判断，

如果宏没有定义则进行编译，否则跳过编译。**include\_guard()** 命令可以为头文件生成这样的宏定义。