## CMake入门04\_使用目标Target

#### 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**

#### 本节主要的内容：

* 什么是Target目标；
* 创建Target的三种方式；
* 依赖关系图；
* 目标属性；
* 什么是传递性使用要求？
* 处理冲突的传播属性；
* 满足伪目标；
* 构建目标；
* 书写定义命令和目标；
* 生成表达式；

#### 什么是Target目标：

如果您曾经使用过 GNU Make，那么您应该已经看到了目标的概念。

本质上，它是构建系统用来将文件列表编译成另一个文件的配方。

它可以是一个.cpp 源码文件编译成一个.o 目标文件，

一组.o 文件打包成一个.a 静态库，以及许多其他组合。

#### 创建目标的三种方式：

* add\_executable()
* add\_library()
* add\_custom\_target()

##### add\_executable命令：

在CMake中，add\_executable用于将源代码文件编译成可执行文件。

社交网站的手机截图

描述已自动生成

##### add\_library命令：

在CMake中，add\_library（添加库）函数用于创建静态或共享库。静态库是在链接时与目标文件一起链接的库，共享库是动态链接库，它在程序运行时加载。

电脑萤幕的截图

描述已自动生成

##### add\_custom\_target命令：

在CMake中，add\_custom\_target命令被用于创建一个自定义目标，其又被用于依赖其它目标或命令，

类似于makefile中的本地目标。它不会产生任何构建输出，而只是定义一个可以触发其它命令或目标的构建行为。

电脑萤幕的截图

描述已自动生成

#### 依赖关系图：

##### 1.图示依赖关系图

图示

描述已自动生成

##### 2.实例代码

文本

中度可信度描述已自动生成

这里需要添加一个依赖关系：



checksum是指对数据进行校验的过程，以确保数据的完整性和准确性。

add\_dependencies函数用于指定一个或多个目标的依赖关系。

例如，如果我们有一个名为checksum的目标，它依赖于另一个名为terminal\_app和gui\_app的目标，我们可以使用以下命令指定依赖关系：

add\_dependencies这将告诉CMake在构建checksum之前先构建terminal\_app和gui\_app。

这在处理复杂的构建系统时非常有用，其中有多个目标需要按特定顺序构建

在上面的示例中，如果我们没有指定add\_dependencies，

则CMake可能会在构建gui\_app或是terminal\_app之前构建checksum，这可能会导致构建失败。因此，使用add\_dependencies可以确保正确的构建顺序。

##### 3.解释上一节课的疑问

**为什么我们可以先声明target\_link\_libraries()在声明add\_library之前？**

我们使用 target\_link\_libraries() 命令将我们的库与可执行文件连接起来。

没有它，可执行文件的编译将因未定义的符号而失败。

您是否注意到我们在实际声明任何库之前调用了此命令？

当 CMake 配置项目时，它会收集有关目标及其属性的信息——它们的名称、依赖项、源文件和其他详细信息。

解析所有文件后，CMake 将尝试构建依赖关系图。

与所有有效的依赖图一样，它们是有向无环图。这意味着哪个目标依赖哪个有明确的方向，并且这种依赖不能形成循环。

当我们在构建模式下执行 cmake 时，生成的构建系统将检查我们定义的顶级目标并递归构建它们的依赖项。

##### 4.构建流程

1. 从顶部开始，构建第 1 组中的两个库。
2. 当计算库和绘图库完成后，构建第 2 组 – GuiApp 和 TerminalApp。
3. 建立校验和目标；运行指定的命令行生成校验和（cksum是一个Unix校验和工具）

##### 5.可视化依赖关系

CMake 有一个很棒的模块可以生成 dot/graphviz 格式的依赖图。并且它支持内部和外部依赖！

图形用户界面, 文本

中度可信度描述已自动生成

该模块将生成一个文本文件，我们可以将其导入 Graphviz 可视化软件，该软件可以渲染图像或生成 PDF 或 SVG 文件

图示

描述已自动生成

#### 目标属性；

在CMake中，Target properties指的是一些对于给定target的属性或特性，包括以下几个方面

* 编译选项（Compile options）
* 链接选项（Link options）
* 编译特性（Compile features）
* 包含目录（Include directories）
* 链接库目录（Link directories）
* 预定义宏（Predefined macros）
* 编译定义（Compile definitions）
* 链接选项（Link options）
* 目标输出名称（Target output name）
* 目标类型（Target type）
* 等等

设置和获取目标属性：

图形用户界面, 文本, 应用程序

描述已自动生成

举例：

文本

描述已自动生成

#### 什么是传递性使用要求？

举个例子，如果目标A使用目标B，而目标B使用目标C，那么目标C是目标A的传递性使用要求。

如果目标C对目标D有库依赖关系，那么目标A也会依赖于目标D，即使它没有在使用要求中明确列出。

总的来说，传递性使用要求是CMake中的一个重要功能，它允许开发人员轻松管理项目中目标之间的复杂依赖关系。

图形用户界面, 文本, 应用程序

描述已自动生成

target\_link\_libraries() 函数是用于将目标和库文件链接起来的函数。它的语法如下：

图形用户界面, 文本

描述已自动生成

target\_compile\_definitions命令，target\_compile\_definitions命令用于为目标添加编译器定义

图形用户界面, 文本, 应用程序

描述已自动生成

#### 处理冲突的传播属性；

在 CMake 中，如果一个 target 依赖于其他多个 target，那么这个 target 所拥有的某些属性，

例如 include目录、编译选项等，可能会受到多个依赖项的冲突，

从而导致编译错误或不正确的行为。解决这些冲突的方法是使用 target\_<prop>\_xxx() 系列函数分别为不同的依赖项设置属性。

图形用户界面, 文本

描述已自动生成

#### 满足伪目标；

在CMake中，伪目标是一种不会自己构建任何内容但作为方便的方式来分组命令和其他目标的方法。一些伪目标的例子包括:

* ALL: 此目标指定在运行“make all”命令时需要构建的所有目标。它通常包括项目中的所有其他目标。
* BUILD: 这是指定要构建的所有目标的另一种方法，但在不同的上下文中使用。这种伪目标允许您设置应用于项目中所有目标的属性。
* CLEAN: 此目标指定在运行“make clean”命令时应清理的所有文件和目录。通常，这包括在构建过程中生成的所有文件。
* TEST: 此目标指定在运行“make test”命令时应运行的所有测试。它通常包括项目中的所有单元测试。
* INSTALL: 此目标指定在运行“make install”命令时应复制到安装目录的所有文件和目录。

使用这些伪目标可以更轻松地管理和组织您的CMake项目。

##### 引入目标；

图形用户界面, 文本, 应用程序

描述已自动生成

##### 别名目标；

图形用户界面, 应用程序

描述已自动生成

##### 3.接口库；

图形用户界面, 应用程序

描述已自动生成

#### 构建特定目标；

当 CMake 生成构建系统时，它会将列表文件从 CMake 语言“编译”为所选构建工具的语言；

也许它会为 GNU Make 创建一个 Makefile。

这样的 Makefile 有它们自己的目标——其中一些是列表文件目标的直接转换，而另一些则是隐式创建的。

一个这样的构建系统目标是 ALL，CMake 默认生成它以包含所有顶级列表文件目标

，例如可执行文件和库（不一定是自定义目标）。 ALL 是在我们运行 cmake --build <build tree> 而不选择具体目标时构建的。

您可能还记得第一章，您可以通过将 --target <name> 参数添加到前面的命令来选择一个。

某些可执行文件或库可能并非在每个构建中都需要，但我们希望将它们保留为项目的一部分，以备不时之需。

为了优化我们的默认构建，我们可以像这样从 ALL 目标中排除它们：

图形用户界面, 应用程序

描述已自动生成

#### 编写定义命令；

##### add\_custom\_command()

向生成的生成系统添加自定义生成规则

第一个签名用于添加自定义命令以产生输出

文本

描述已自动生成

第二个签名向目标(如库或可执行文件)添加自定义命令。这对于在构建目标之前或之后执行操作非常有用。

该命令成为目标的一部分，仅在构建目标本身时执行。如果目标已经构建，则该命令将不会执行

文本

描述已自动生成

##### add\_custom\_target()

添加一个没有输出的目标，以便始终构建它。

文本

描述已自动生成

##### 举例说明：

图形用户界面, 应用程序, Teams

描述已自动生成

每次构建前先清理之前的构建文件：

图形用户界面, 应用程序

描述已自动生成

#### 生成表达式；

CMake生成器表达式是CMake的语法扩展，允许您基于目标系统、配置类型或构建工具链等内容指定构建设置。

这些表达式主要用于**target\_compile\_options()**和**target\_link\_options()**命令的上下文中，但也可用于其他CMake命令中。

电脑萤幕的截图

描述已自动生成