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

**插图以及内容来自书籍《Modern CMake For C++》**

**辅助工具：ChatGPT**

**CMake入门01-CMake是什么？**

本节主要的内容：

* CMake的基本概念以及优势；
* CMake的核心概念和处理流程。
* CMake的主要使用的命令；
* CMake支持的生成器有哪些；
* 各个IDE对CMake的支持情况？

文本

描述已自动生成

CMake的主要作用：

* 生成可以执行文件和库文件
* 管理依赖；（第三库，FFMPEG，SDL2）
* 测试；CTest
* 安装；
* 打包；CPack
* 自动生成文档；（类似于[Swagger](https://swagger.io/)）

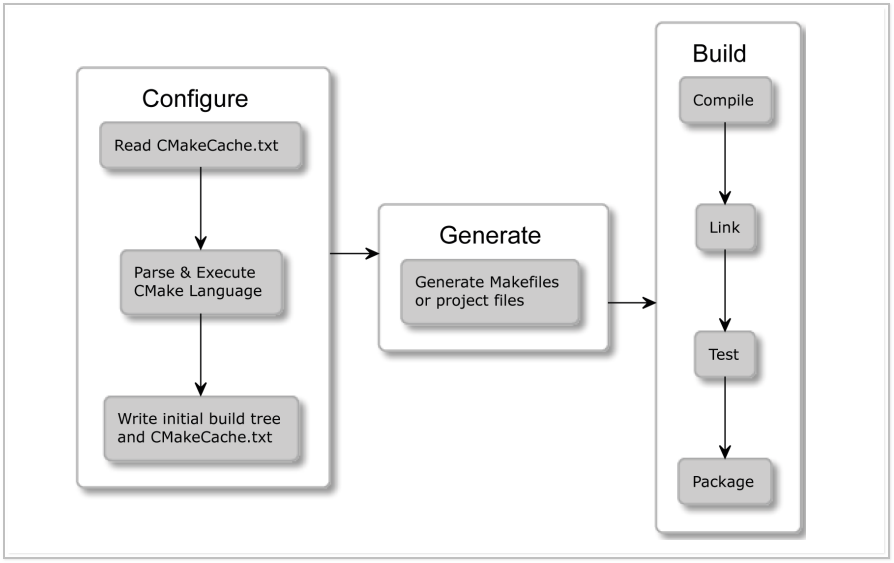
CMake的主要优点：

* 管理复杂的工程文件结构。（嵌套（子目录），外部依赖（第三方库），内部依赖，IDE虽然可以做，不过很多不支持跨平台）
* 支持现代的编译器和工具链；
* 跨平台构建，支持Windows，Linux，MacOS和Cygwin；
* 可以为主流的IDE生成工程文件（VS，Xcode，Eclipse CDT，CLion）
* 可维护性和抽象；（可重用）
* 有大量的程序基于CMake构建，可以非常方便的集成到自己的项目中；
* 简化构建过程，CMake支持测试，打包，安装

CMake一个构建工具，经过20多年的发展，已经是事实上的行业标准；

CMake处理分为三个步骤：

1. 配置；
2. 生成构建文件；
3. 构建；



配置（Configure）：用来描述怎么构建过程，根据CMakeLists.txt文件所包含的指令和配置选项，为项目生成相应的生成器、

编译系统或工程文件的过程（比如：确定系统和开发环境，描述项目名称，源文件，头文件，库，以及依赖等等）

生成器(Generators)：CMake用来生成特定编译器或集成开发环境(IDE)项目文件等工具的插件。

构建（Build）: 把源代码转为可执行文件或是库的过程，其中包括了（读取编译配置，编译，链接，测试，安装等）

CMake 本身是无法构建任何东西——它依赖于系统中的其他工具来执行实际的编译、链接和其他任务。

您可以将其视为构建过程的协调器：它知道需要完成哪些步骤，最终目标是什么，以及如何为找到合适的脚本和材料

最简单的CMakeLists.txt的结构：

1. 生成可执行文件：

图形用户界面, 文本, 应用程序, 聊天或短信

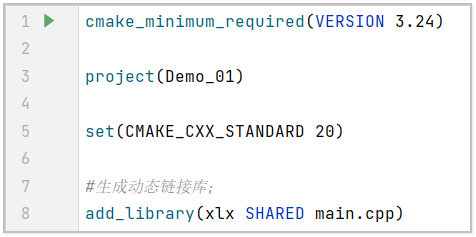
描述已自动生成

1. 生成静态链接库：

图形用户界面, 文本, 应用程序

描述已自动生成

1. 生成动态链接库：



确保电脑已经安装了cmake：

检查cmake是否存在：

cmake –version

查看帮助文档：

cmake -h

* -S <path-to-source> = Explicitly specify a source directory.
* -B <path-to-build> = Explicitly specify a build directory.
* --build <dir> = Build a CMake-generated project binary tree.
* -G <generator-name> = Specify a build system generator.

最重要的两个命令：

表格

描述已自动生成

其中-S指定要构建的源码文件路径；

其中-B指定要生成的Build Tree路径；

* 外部构建，源码和构建文件分开；（推荐，这样可以保存源码不被污染）
* 内部构建：

.. 两个点表示上级目录；

. 一个点表示当前目录；

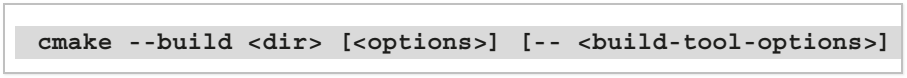
比如定义编译Release或是debug版本；



cmake -S . -B build -D CMAKE\_BUILD\_TYPE=Release

cmake -S . -B build -D CMAKE\_BUILD\_TYPE=Debug

构建命令：



注意这里的dir是指build tree，比如上个命令的build目录（也就是上一步生成构建文件的-B参数；）

指定参数：比如构建Release或是Debug版本；

cmake --**build** . --config Debug

cmake --**build** . --config Release

总的来说，--config选项只在使用Visual Studio生成器时有用。

如果你使用其他生成器，

可以在命令中直接指定CMAKE\_BUILD\_TYPE变量来选择Release或Debug版本，无需使用--config。

使用不同的生成器：

-G Ninja

-G MinGW Makefiles

-G “Visual Studio 17 2022”

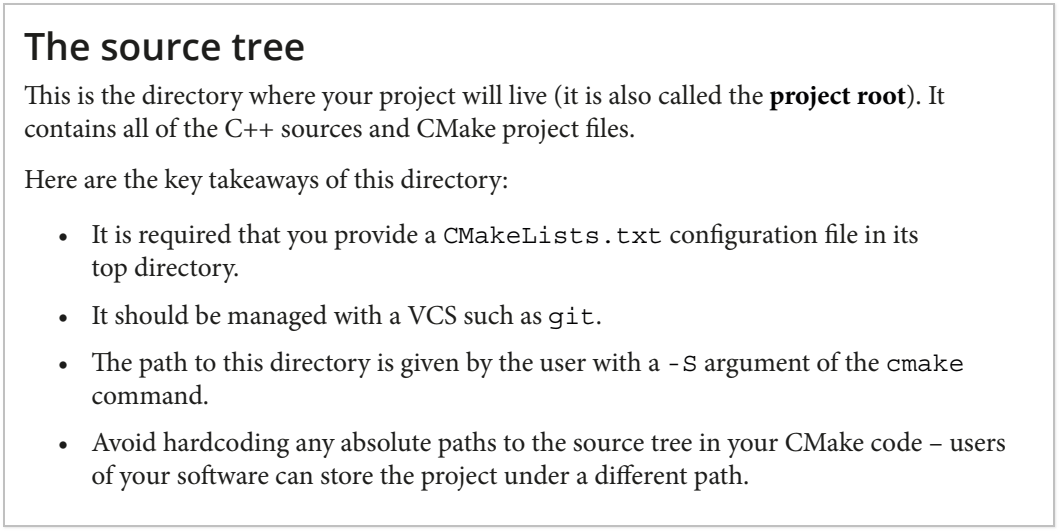
许多教程建议生成阶段后直接运行 GNU Make：make，这是 Linux 和 macOS 的默认生成器

但是，建议一直用cmake，因为它独立于生成器并且在所有平台上都受支持。

因此，我们无需担心我们应用程序的每个用户的确切环境。

好久比TypeScript和JavaScript的关系：

Cmake->makefile文件->make；



文本

描述已自动生成

文本

描述已自动生成

CMake支持多种生成器，包括:



主流IDE对cmake的支持：

* QT Creator（QT6）开始使用cmake构建；
* Clion 使用cmake构建；（Clion支持调试）
* Visual studio支持cmake构建；
* Eclipse CDT

CMake本身不会编译，它依赖于系统的其他工具完成实际的编译，链接以及其他任务；

图形用户界面, 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

什么是Ninja生成器，它有什么优点：

