FreedomBytes

2023

CMake: An introduction to build, test and package software

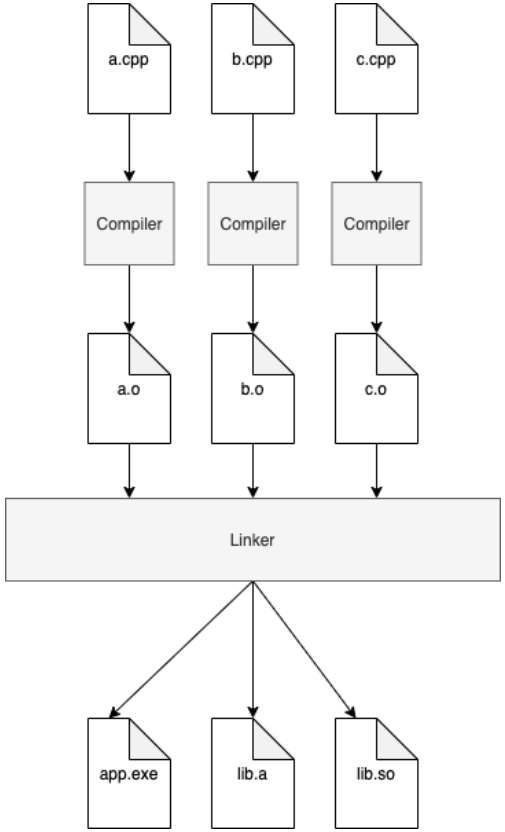
- La domanda fondamentale: Che cos’è CMake?
- Facciamo ordine: da dove cominciare?
- Compilazione programmi C/C++: make files e standard convetions

Che cos’è CMake?

- CMake is the **de-facto standard** for building C++ code.
- CMake is an extensible, open-source system that manages the **build process** in an operating system and in a compiler-independent manner.
- CMake is **a collection of** open-source and cross-platform **tools** used to **build** and **distribute** software.
- CMake is known as a **meta build system**. It doesn't actually build your source code: instead, it generates native project files for the target platform.

Iniziamo: Build

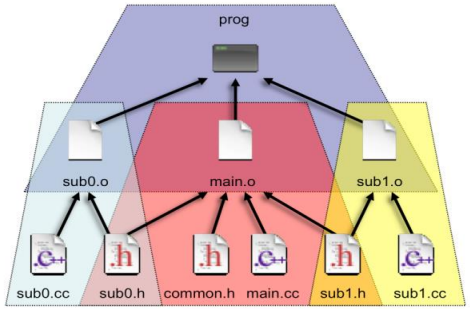
- **Progetto: insieme di file di testo**
- **Compilatore: trasforma ogni file in codice oggetto**
- **Linker: collega tutto insieme**
- **Risultato:**
  - Eseguibile
  - Libreria statica
  - Libreria dinamica



Build in practice

```
> g++ -c a.cpp -o a.o

> g++ -c b.cpp -o b.o

> g++ -c c.cpp -o c.o

> g++ -c main.cpp -o main.o

> g++ a.o b.o c.o main.o –o main

> ./main
```

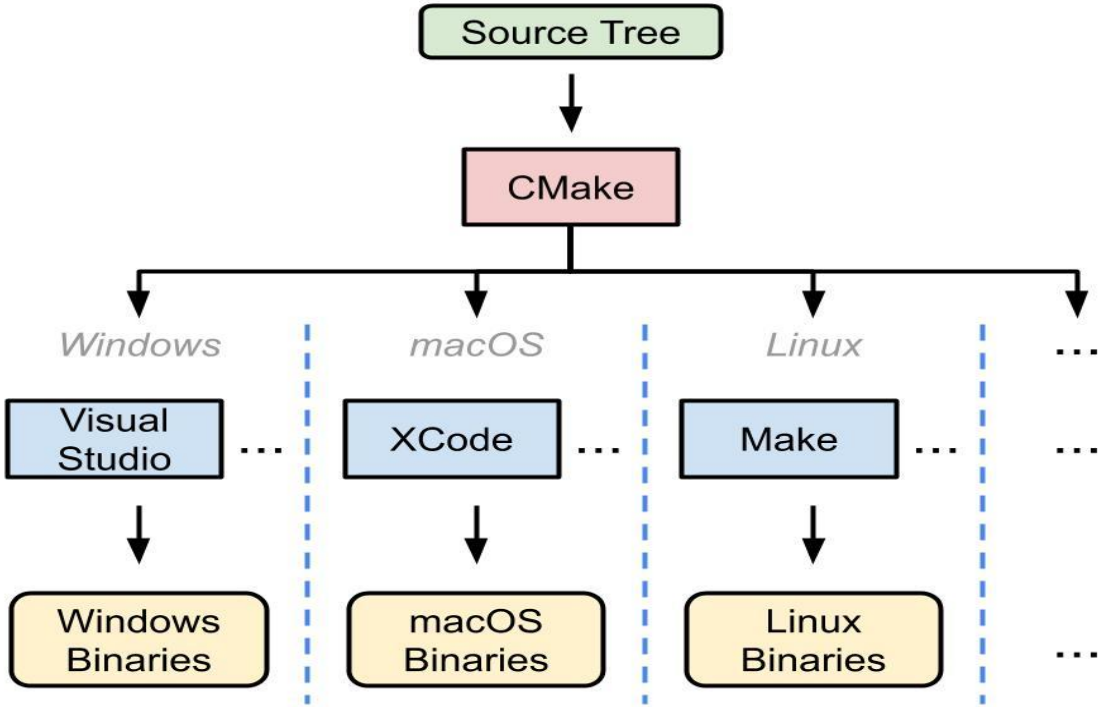
MakeFile



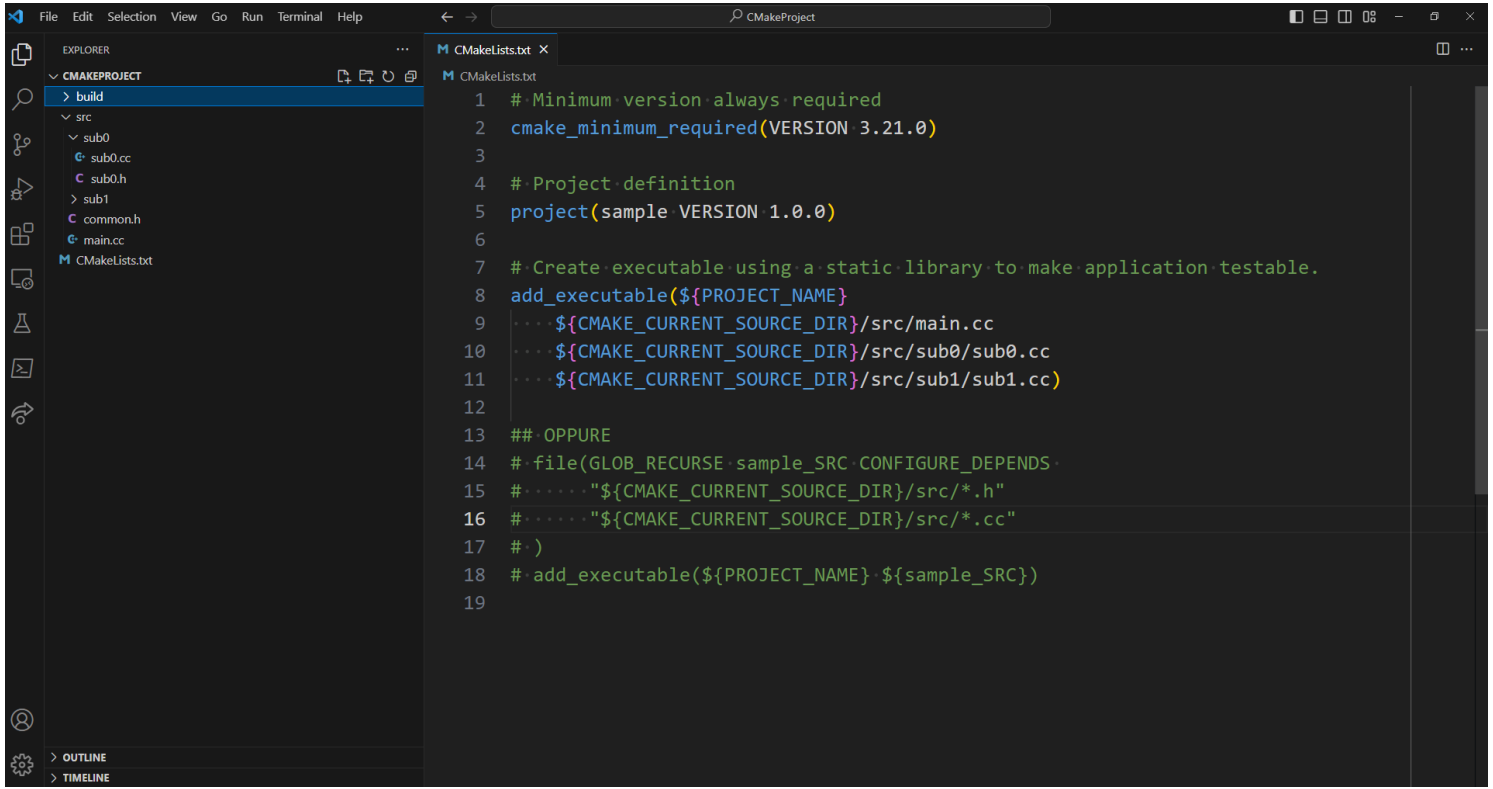

```
all: sub0.o sub1.o main.o
        mpicxx sub0.o sub1.o main.o -o prog
sub0.o: sub0.cc sub0.h
        mpicxx -c sub0.cc
sub1.o: sub1.cc sub1.h
        mpicxx -c sub1.cc
main.o: main.cc common.h sub0.h sub1.h
        mpicxx -c main.cc
clean:
        rm *.o prog
```

- Difficile da manutenere
- Dipendenza dal compilatore: se non con boilerplate complessi
- Dipendenza dal Sistema: unix
- Linkare librererie esterne può diventare complicato

Arriviamo a CMake



Primo Esempio

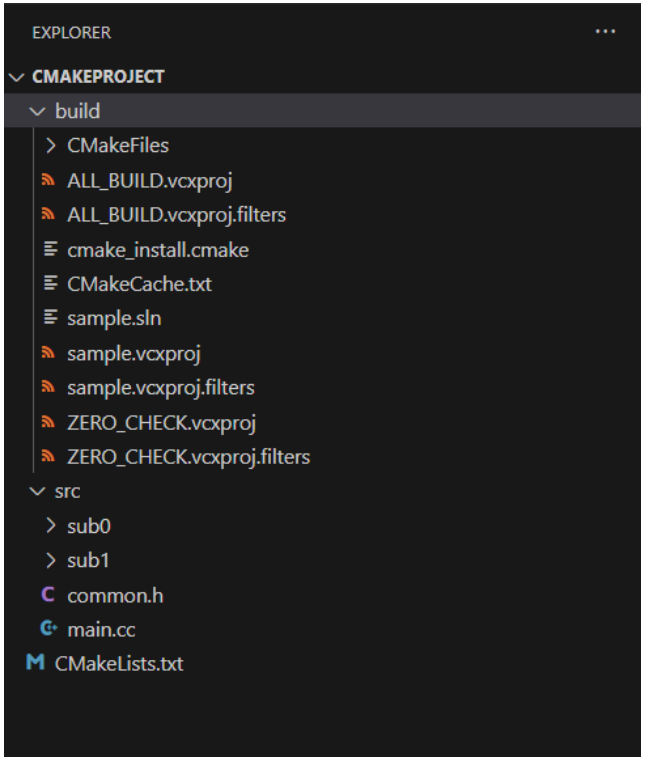


Comandi

Generare i file nella cartella attuale

> cmake .

Generare i file nella cartella ‘build’:
> cmake –B build



ATTENZIONE

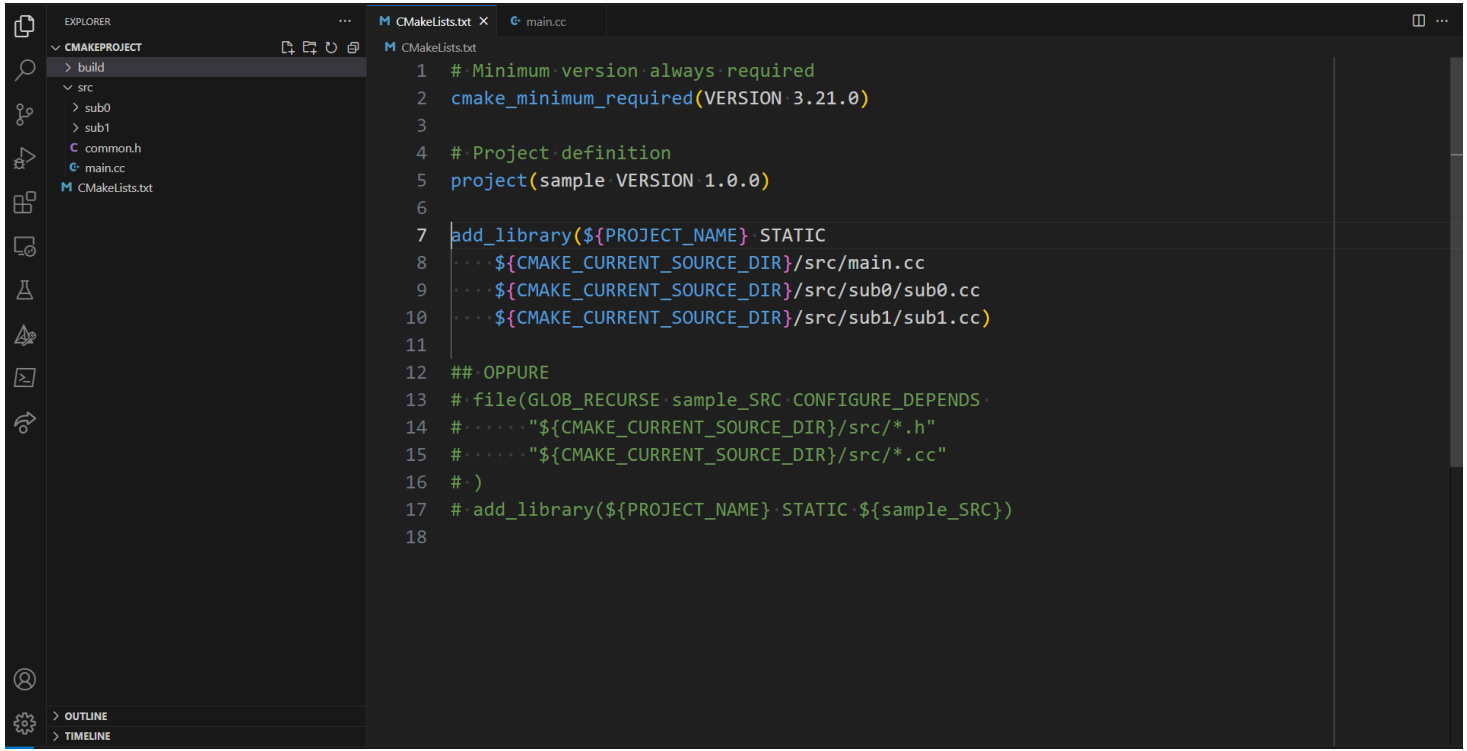
- Cosa succede se modifico un file?
- Cosa succede se aggiungo un file?
- Cosa succede se elimino un file?

Target

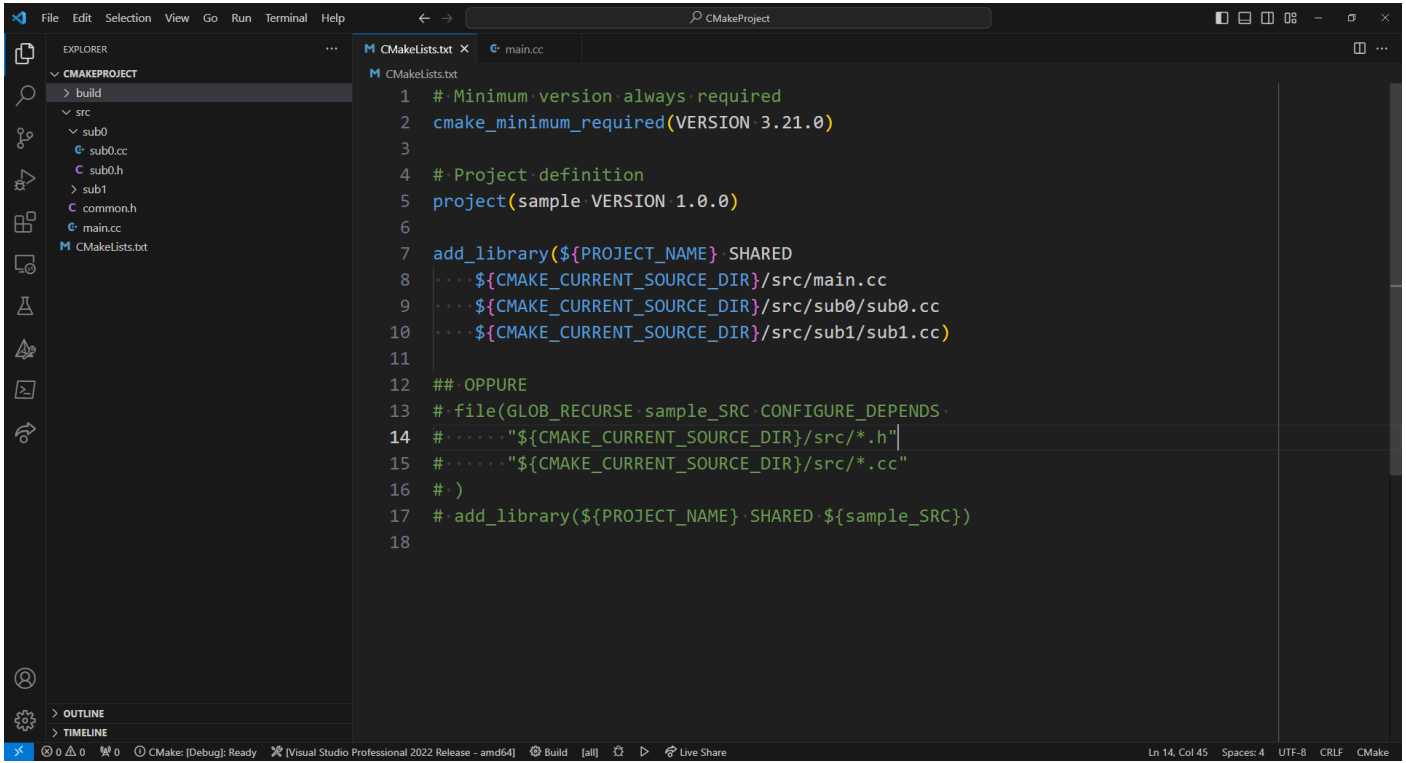
- Eseguibile
- Libreria Statica
- Libreria Dinamica
- Libreria Header Only

- add_executable(<name>)
- add_library(<name> STATIC)
- add_library(<name> SHARED)
- add_library(<name> INTERFACE)

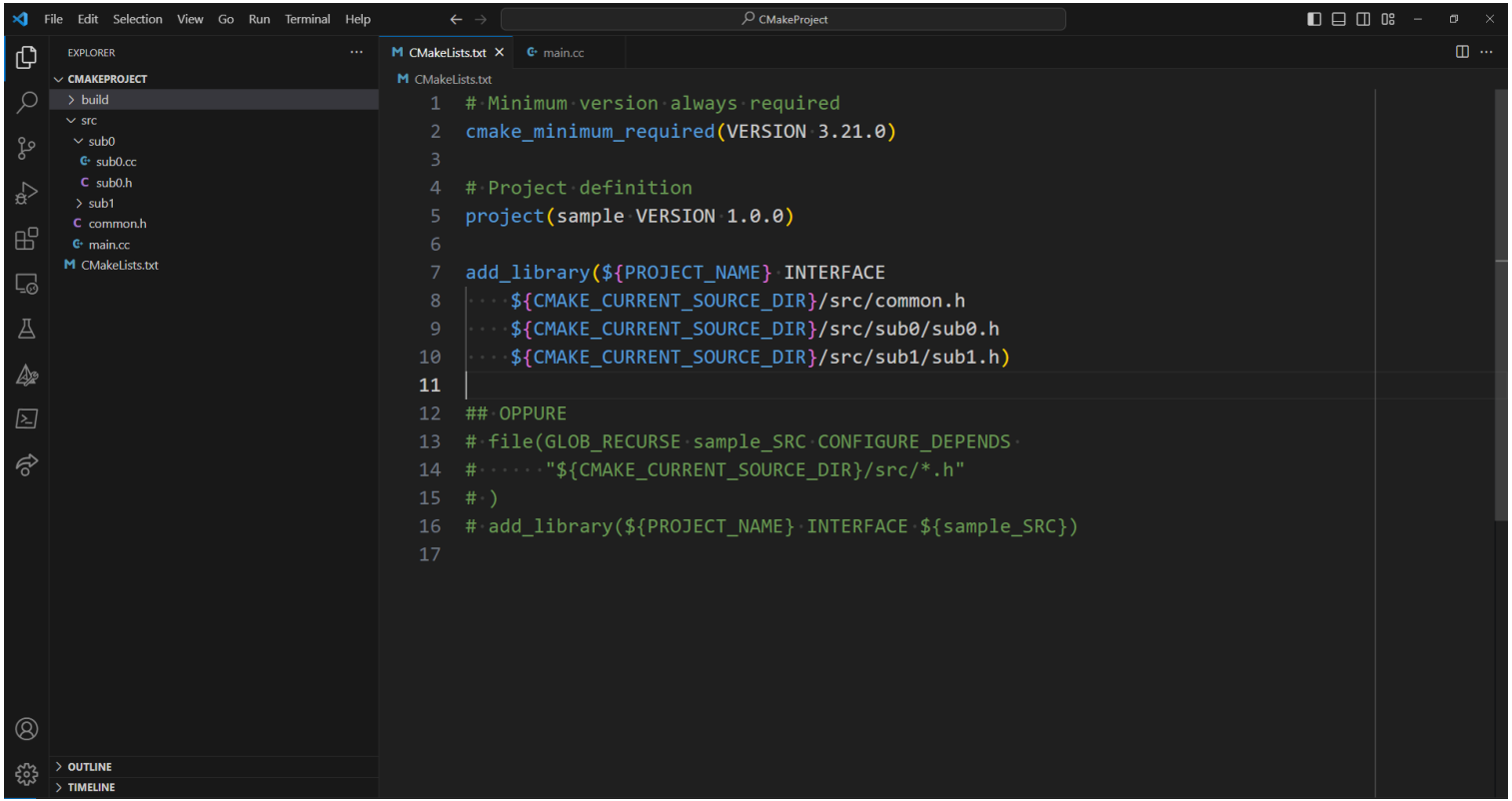
Libreria Statica



Libreria Dinamica



Libreria Header Only



Usare una libreria: cosa serve

- La libreria da linkare: bisogna dirlo al linker
- La cartella dove andare a cercare la libreria da linkare
- La cartella dove andare a cercare gli header

The DIRECTORY form installs contents of one or more directories to a given destination. The directory structure is copied verbatim to the destination.

Include header: dobbiamo dirlo noi

- **Modo 1:**

```
# The include directories are added to the INCLUDE_DIRECTORIES directory
# property for the current CMakeLists file. They are also added to the
# INCLUDE_DIRECTORIES target property for each target in the current CMakeLists
# file. The target property values are the ones used by the generators.
include_directories(PUBLIC "include")
# The DIRECTORY form installs contents of one or more directories to a given
# destination. The directory structure is copied verbatim to the destination.
install(DIRECTORY "include" DESTINATION .)
```

- **Modo 2: Generator properties**

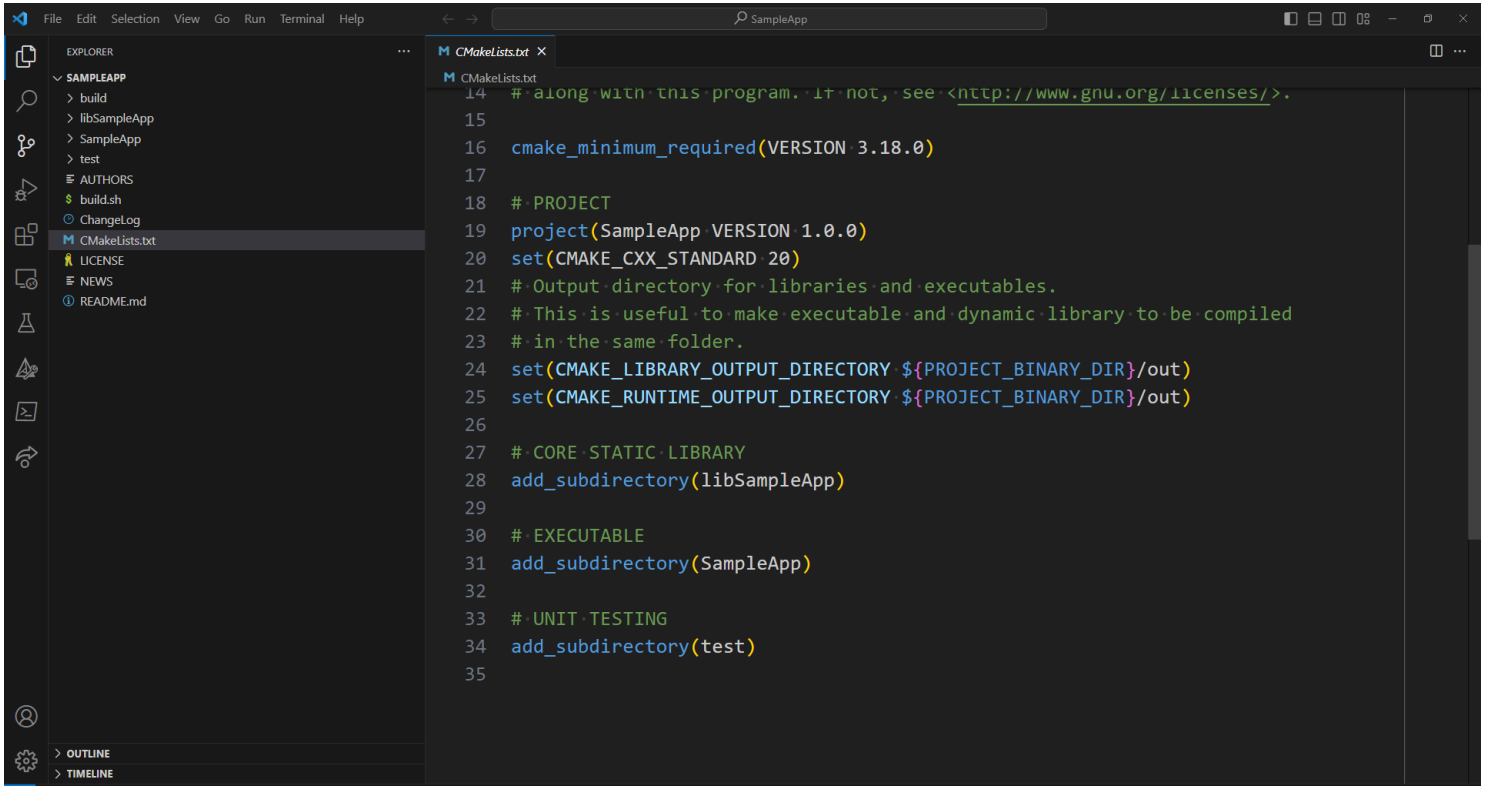
```
target_include_directories(${PROJECT_NAME} PUBLIC
    "$<BUILD_INTERFACE:${CMAKE_CURRENT_LIST_DIR}/include>"
    "$<INSTALL_INTERFACE:${CMAKE_INSTALL_INCLUDEDIR}>")
```

Linkare una libreria

```
target_link_libraries(
   <target_name>
   PRIVATE|PUBLIC|INTERFACE <library_name1>
   PRIVATE|PUBLIC|INTERFACE <library_name1>
   …

   PRIVATE|PUBLIC|INTERFACE <library_nameN>)
```

- **PRIVATE: la dipendenza e' visible solo nei file sorgenti**
- **PUBLIC: la dipendenza e' visible sia nei file sorgenti che negli header**
- **INTERFACE: la dipendenza e' visible solo nei file header**

Struttura Applicazione



libSampleApp

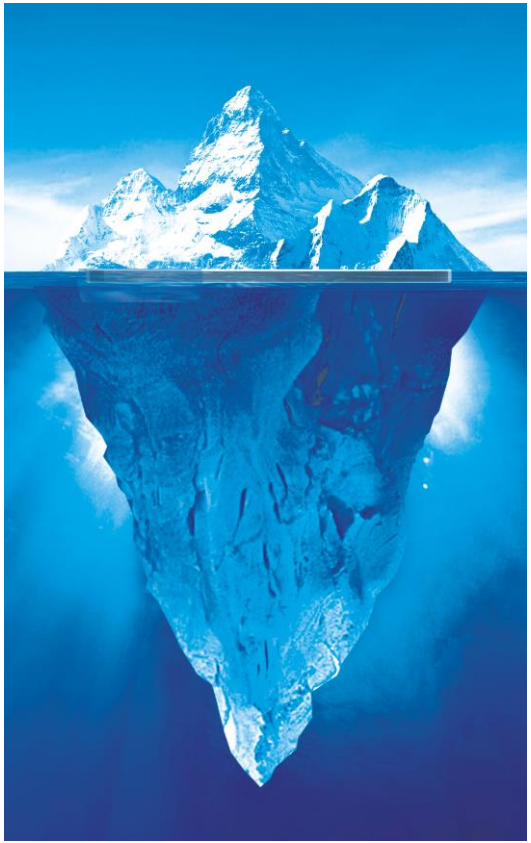
```
# Create application core as static library
add_library(lib${PROJECT_NAME} STATIC "")

file(GLOB lib${PROJECT_NAME}_SRC CONFIGURE_DEPENDS "src/*.cc")

target_sources(lib${PROJECT_NAME} PRIVATE ${lib${PROJECT_NAME}_SRC})

# Public interface
target_include_directories(lib${PROJECT_NAME} PUBLIC
    "$<BUILD_INTERFACE:${CMAKE_CURRENT_LIST_DIR}/include>"
     "$<INSTALL_INTERFACE:${CMAKE_INSTALL_INCLUDEDIR}>")
```

SampleApp

```
add_executable(${PROJECT_NAME})

file(GLOB ${PROJECT_NAME}_SRC CONFIGURE_DEPENDS "*.cc")

target_sources(${PROJECT_NAME} PRIVATE ${${PROJECT_NAME}_SRC})

target_link_libraries(${PROJECT_NAME} PRIVATE lib${PROJECT_NAME})
```

SampleAppTest

```
add_executable(${PROJECT_NAME}Test)

file(GLOB ${PROJECT_NAME}_TEST_SRC CONFIGURE_DEPENDS "src/*.cc")

target_sources(${PROJECT_NAME}Test PRIVATE ${${PROJECT_NAME}_TEST_SRC})

target_link_libraries(${PROJECT_NAME}Test PRIVATE lib${PROJECT_NAME})
```



- If/else
- Loop
- Include cmake files
- Package system
- Etc..

Ricapitolando

- Multipiattaforma
- Semplifica operazioni laboriose: linking
- Gestione del Progetto as folder
- Si integra con diversi IDE ed editor
- Ben supportato da conan

Domande?