C++ (cmake) Cheat Sheet

by blakecromar via cheatography.com/218070/cs/48226/

add_executable

```
add_executable(executable_name,
main.cpp,
source _fi le.cpp
...
)
```

Defines a new executable build target named <executable_name>.
Lists the source files that will be compiled and linked to produce the binary.
The target name can later be referenced by target_link_libraries, target_include_directories, etc.

add_library vs add_executable

```
add_library(library_name
        sou rce _fi le.cpp
       ...
)
add_ex ecu tab le( exe cut abl -
e_name
        mai n.cpp
       ...
)
```

An executable has a main() function, can be run directly, and is always the end of the dependency chain.

A library has no main() function, cannot run on its own, and exists only to be linked against by other targets.

PRIVATE vs PUBLIC only matters for libraries — executables are always PRIVATE.

Choosing PRIVATE vs PUBLIC for Libraries

```
target_link_libraries(my_library
        PUBLIC qt5::Core #
appears in my headers, must be
passed on
        PRIVATE intern al_ -
helper # only in my .cpp files,
stays hidden
)
```

Ask yourself: do my header files expose this dependency to the outside world?

If the dependency appears in your headers, anything linking against you will also need it

target_include_directories

```
target_include_directories(executable_name
PRIVATE
includ e_d ire ctory
...
)
```

Specifies include directories for a given target, making headers in those directories available via `#include`.
`PRIVATE` means the directories are only used when building this target, not propagated to dependents.
Other scope options are `PUBLIC` (propagated to dependents) and `INTERFACE` (only propagated, not used by the target itself).

CMake File Traversal

```
# root CMakeLists.txt
includ e(c mak e/o pen cv.c -
make) # register libraries first
add_su bdi rec tor y(s rc/ lib -
s/l ogger) # process
logger /CM ake Lis ts.txt
add_su bdi rec tor y(s rc/ -
tests) # process tests/ CMa -
keL ist s.txt
```

The root CMakeLists.txt is the entry point and orchestrates the entire build.

include() pulls in .cmake files early to register libraries and settings before any targets are defined.

add_subdirectory() tells CMake to go into a folder and process the CMakeLists.txt there before continuing.

Leaf CMakeLists.txt files define targets with add_executable or add_library and have no further add_subdirectory() calls.

CMAKE_CURRENT_SOURCE_DIR

target_link_libraries

```
target_link_libraries(executable_n
PRIVATE
librar y_name
...
)
```

Specifies which libraries an executable or library target should be linked against.

`PRIVATE` means the libraries are only used when building this target, not propagated to dependents.

Without this, the linker would not know where to find the implementations of external code your target depends on.

PRIVATE / PUBLIC

```
target_link_libraries(target_name
        PRIVATE privat e_d epe -
ndency # stays hidden inside
this target
        PUBLIC public _de pen -
dency # passed on to anything
linking against this target
)
```

Controls whether dependencies are propagated to targets that link against this one.

Use PRIVATE when a dependency only appears in .cpp files (hidden inside the implementation), use PUBLIC when a dependency appears in header files (exposed to the outside world).

Executables are always PRIVATE since nothing ever links against them.

— use PUBLIC.

If the dependency only appears in your .cpp files, it stays hidden inside — use PRIVATE.

```
target_include_directories(executable_name
PRIVATE
        ${C MAK E_C URR ENT _SO URC E_DIR}
# this CMakeL ist s.txt's folder
        ${C MAK E_C URR ENT _SO URC E_D -
IR}/.. # one folder up
        ${C MAK E_C URR ENT _SO URC E_D -
IR} /in clude # subfolder
)
```

A built-in CMake variable that always points to the directory containing the currently processed CMakeLists.txt.

Use it to build paths to files relative to the current CMakeLists.txt location.

/.. moves up one directory, making it useful for referencing files in a parent folder.