C++ now

2025

Post-Modern Cmake

From 3.0 to 4.0

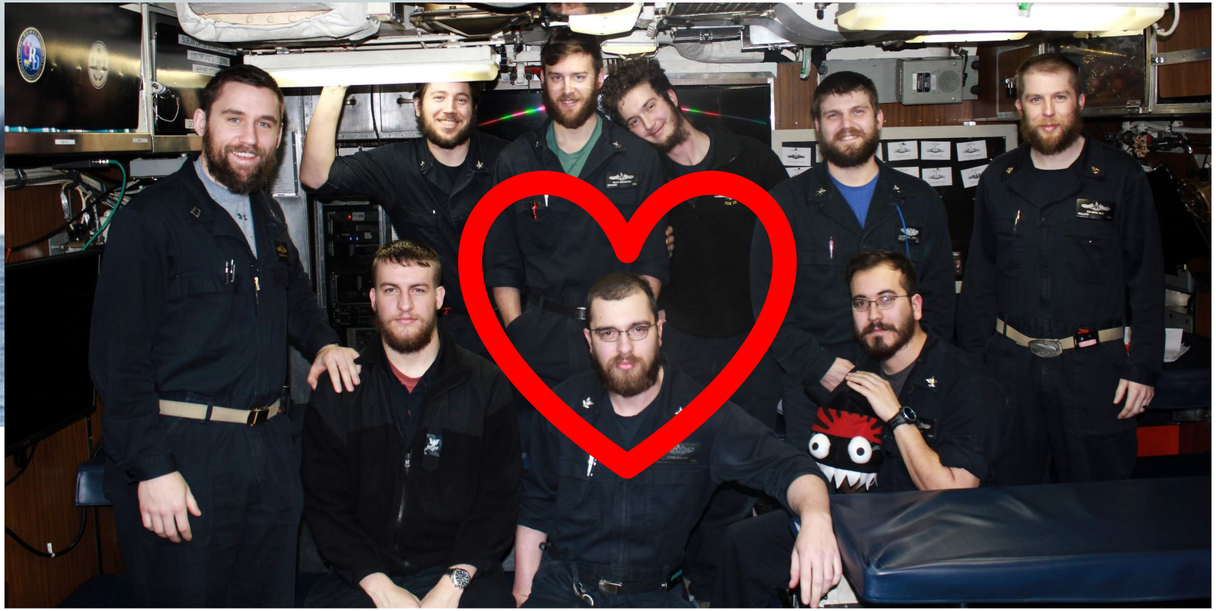
Vito Gamberini

Obligatory Speaker Introduction

Obligatory Speaker Introduction

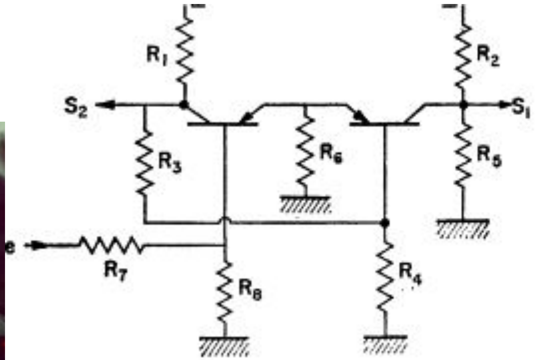
Obligatory Speaker Introduction

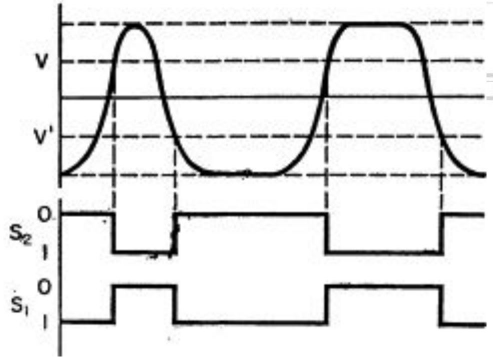
Obligatory Speaker Introduction



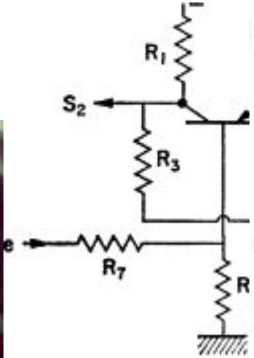
Obligatory Speaker Introduction

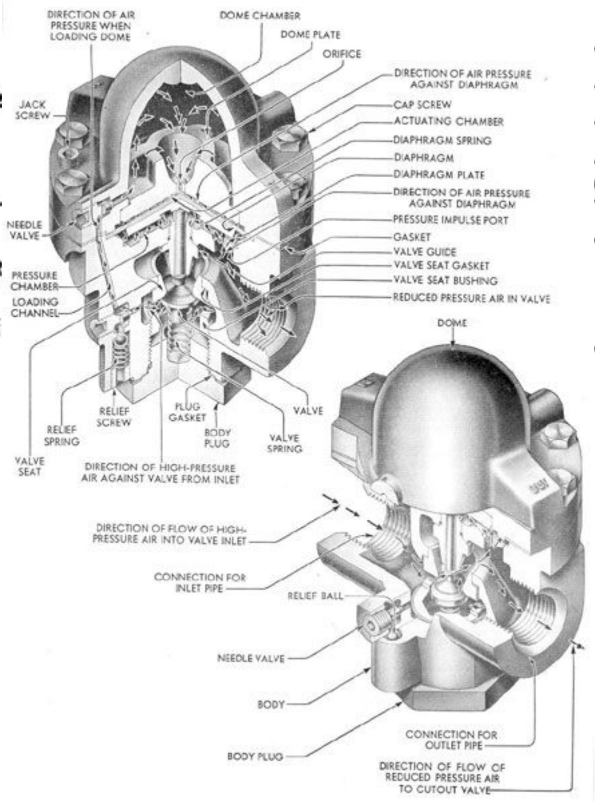
Obligatory Speaker Introduction



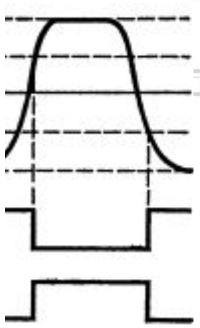


Obligatory Speaker Introduction

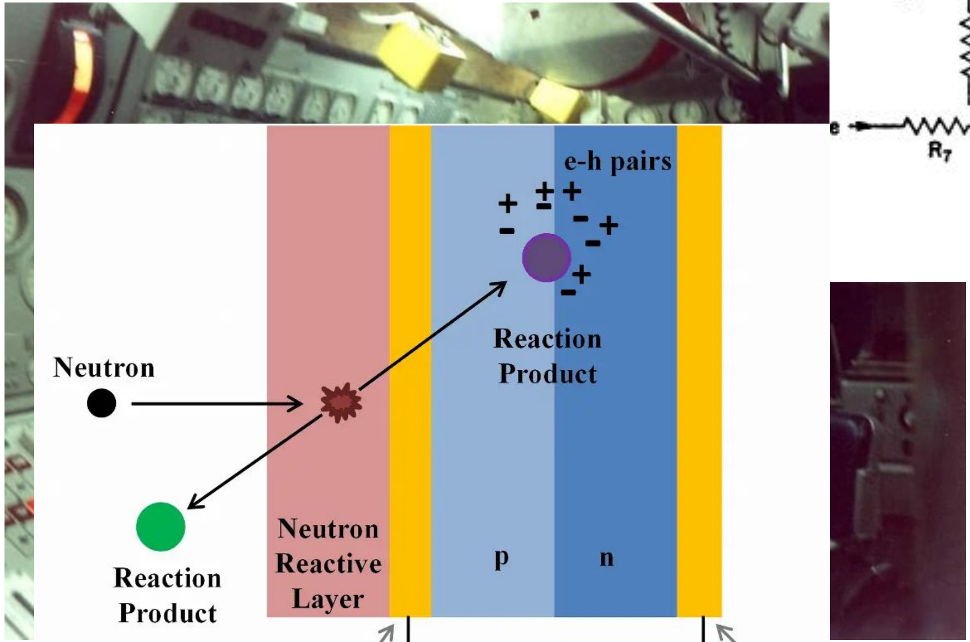


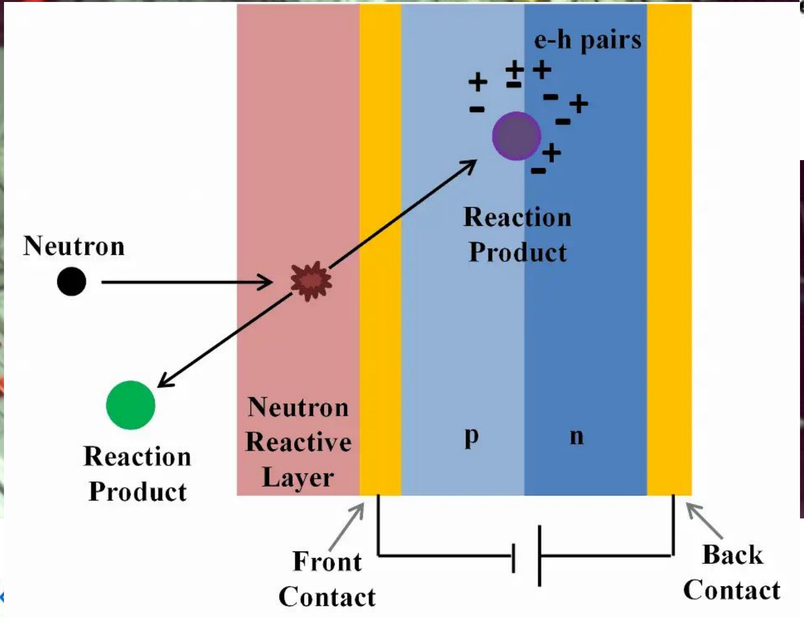


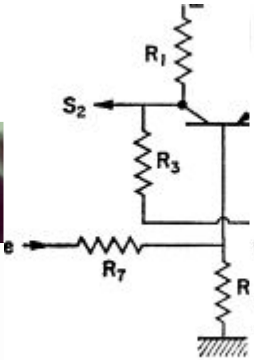
DIRECTION OF AIR PRESSURE WHEN LOADING DOME
DOME CHAMBER
DOME PLATE
ORIFICE
DIRECTION OF AIR PRESSURE AGAINST DIAPHRAGM
JACK SCREW
CAP SCREW
ACTUATING CHAMBER
DIAPHRAGM SPRING
DIAPHRAGM
DIAPHRAGM PLATE
DIRECTION OF AIR PRESSURE AGAINST DIAPHRAGM
NEEDLE VALVE
PRESSURE IMPULSE PORT
GASKET
VALVE GUIDE
VALVE SEAT GASKET
PRESSURE CHAMBER
VALVE SEAT BUSHING
LOADING CHANNEL
REDUCED PRESSURE AIR IN VALVE
DOME
RELIEF SCREW
PLUG GASKET
VALVE
RELIEF SPRING
BODY PLUG
VALVE SPRING
VALVE SEAT
DIRECTION OF HIGH-PRESSURE AIR AGAINST VALVE FROM INLET
DIRECTION OF FLOW OF HIGH-PRESSURE AIR INTO VALVE INLET
CONNECTION FOR INLET PIPE
RELIEF BALL
NEEDLE VALVE
BODY
CONNECTION FOR OUTLET PIPE
BODY PLUG
DIRECTION OF FLOW OF REDUCED PRESSURE AIR TO CUTOUT VALVE

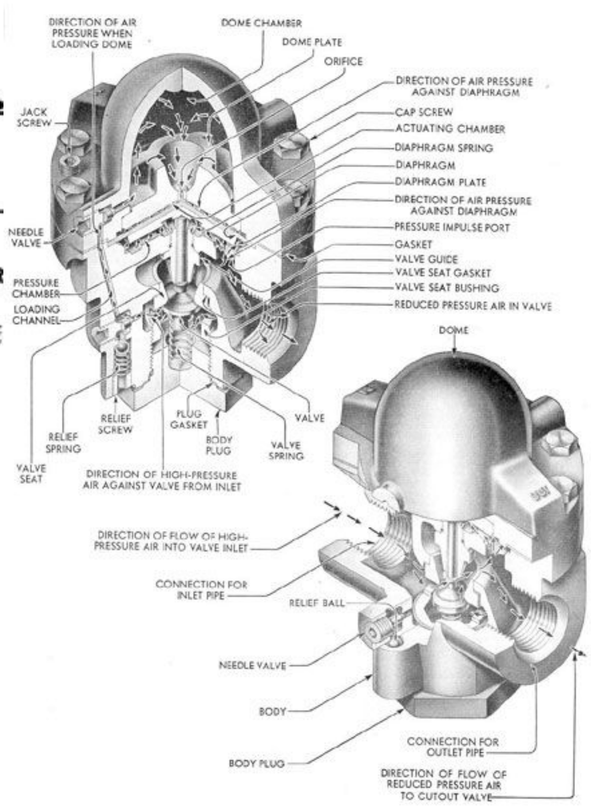


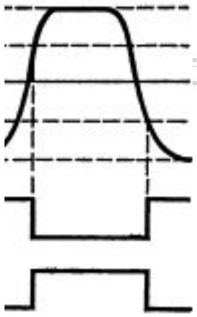
Obligatory Speaker Introduction



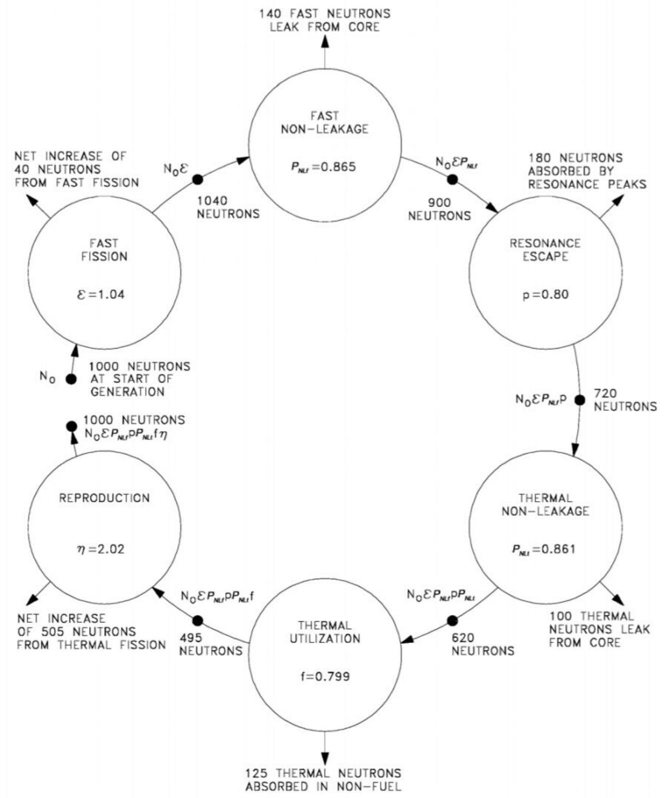





DIRECTION OF AIR PRESSURE WHEN LOADING DOME
DOME CHAMBER
DOME PLATE
ORIFICE
DIRECTION OF AIR PRESSURE AGAINST DIAPHRAGM
JACK SCREW
CAP SCREW
ACTUATING CHAMBER
DIAPHRAGM SPRING
DIAPHRAGM
DIAPHRAGM PLATE
DIRECTION OF AIR PRESSURE AGAINST DIAPHRAGM
NEEDLE VALVE
PRESSURE IMPULSE PORT
GASKET
VALVE GUIDE
VALVE SEAT GASKET
PRESSURE CHAMBER
VALVE SEAT BUSHING
LOADING CHANNEL
REDUCED PRESSURE AIR IN VALVE
DOME
RELIEF SCREW
PLUG GASKET
VALVE
RELIEF SPRING
BODY PLUG
VALVE SPRING
VALVE SEAT
DIRECTION OF HIGH-PRESSURE AIR AGAINST VALVE FROM INLET
DIRECTION OF FLOW OF HIGH-PRESSURE AIR INTO VALVE INLET
CONNECTION FOR INLET PIPE
RELIEF BALL
NEEDLE VALVE
BODY
BODY PLUG
CONNECTION FOR OUTLET PIPE
DIRECTION OF FLOW OF REDUCED PRESSURE AIR TO CUTOUT VALVE

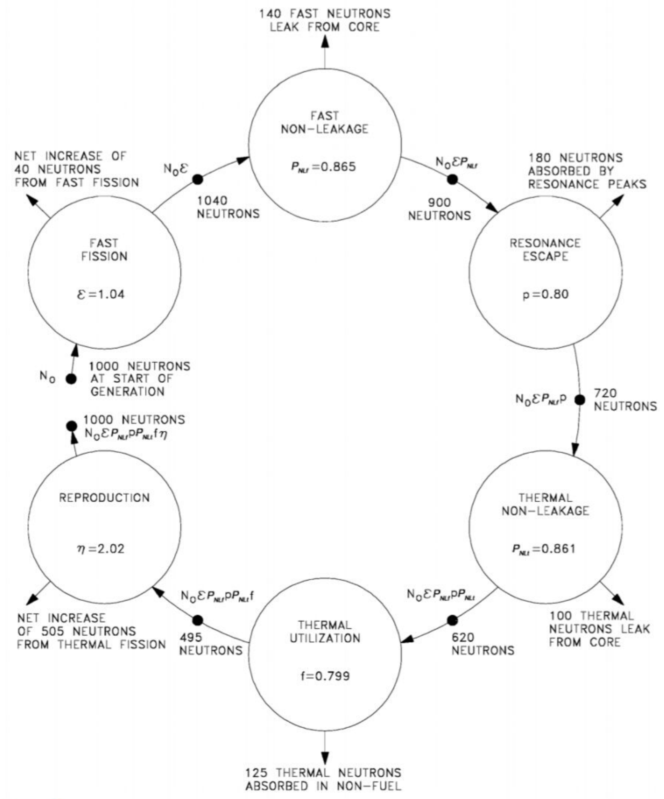


Obligatory Speaker Introduction



140 FAST NEUTRONS LEAK FROM CORE
FAST NON-LEAKAGE
$P_{NLf}=0.865$
$N_0\varepsilon P_{NLf}$
180 NEUTRONS ABSORBED BY RESONANCE PEAKS
NET INCREASE OF 40 NEUTRONS FROM FAST FISSION
$N_0\varepsilon$
1040 NEUTRONS
900 NEUTRONS
FAST FISSION
$\varepsilon=1.04$
RESONANCE ESCAPE
$p=0.80$
N_0
1000 NEUTRONS AT START OF GENERATION
$N_0\varepsilon P_{NLf}p$
720 NEUTRONS
1000 NEUTRONS
$N_0\varepsilon P_{NLf}pP_{NLt}f\eta$
REPRODUCTION
$\eta=2.02$
THERMAL NON-LEAKAGE
$P_{NLt}=0.861$
$N_0\varepsilon P_{NLf}pP_{NLt}f$
$N_0\varepsilon P_{NLf}pP_{NLt}$
NET INCREASE OF 505 NEUTRONS FROM THERMAL FISSION
495 NEUTRONS
THERMAL UTILIZATION
$f=0.799$
620 NEUTRONS
100 THERMAL NEUTRONS LEAK FROM CORE
125 THERMAL NEUTRONS ABSORBED IN NON-FUEL

Obligatory Speaker Introduction



140 FAST NEUTRONS LEAK FROM CORE
FAST NON-LEAKAGE
$P_{NLF}=0.865$
$N_0\varepsilon P_{NLF}$
180 NEUTRONS ABSORBED BY RESONANCE PEAKS
900 NEUTRONS
RESONANCE ESCAPE
p=0.80
$N_0\varepsilon P_{NLF}p$
720 NEUTRONS
THERMAL NON-LEAKAGE
$P_{NLT}=0.861$
100 THERMAL NEUTRONS LEAK FROM CORE
$N_0\varepsilon P_{NLF}pP_{NLT}$
620 NEUTRONS
THERMAL UTILIZATION
f=0.799
125 THERMAL NEUTRONS ABSORBED IN NON-FUEL
$N_0\varepsilon P_{NLF}pP_{NLT}f$
495 NEUTRONS
NET INCREASE OF 505 NEUTRONS FROM THERMAL FISSION
REPRODUCTION
$\eta=2.02$
1000 NEUTRONS $N_0\varepsilon P_{NLF}pP_{NLT}f\eta$
N_0
1000 NEUTRONS AT START OF GENERATION
FAST FISSION
$\varepsilon=1.04$
NET INCREASE OF 40 NEUTRONS FROM FAST FISSION
$N_0\varepsilon$
1040 NEUTRONS

CMake

autoconf

ninja
BUILD SYSTEM

kitware

Obligatory Cop Outs

- I work for Kitware
- I do not *speak* for Kitware

Obligatory Cop Outs

- I work for Kitware
- I do not *speak* for Kitware
- I work for the CMake team
- You get the idea

Obligatory Cop Outs

- I work for Kitware
- I do not *speak* for Kitware
- I work for the CMake team
- You get the idea
- No promises, no commitments
- I reserve the right to be wrong

Agenda

- Pragmatic CMake Usage
- Latest Features: C++20 Module Support
- New Stuff On the Horizon: CPS

C++ now

2025

Pragmatic CMake

How to Avoid Headaches with Simple CMake

Bret Brown

C++ now

2025

import CMake; // Mastering C++ Modules

Marching Towards Standard C++ Dependency Management

Bill Hoffman

C++ now

2025

CPS in Cmake

Marching Towards Standard C++ Dependency Management

Bill Hoffman

Eras of CMake

Eras of CMake

- CMake 1: Primordial Flag Soup
- CMake 2: World Domination
- CMake 3: I hope you like target_* commands
- CMake 4: ???

CMake 1: Primordial Flag Soup

```
PROJECT(CMake)
SUBDIRS(Source)
```

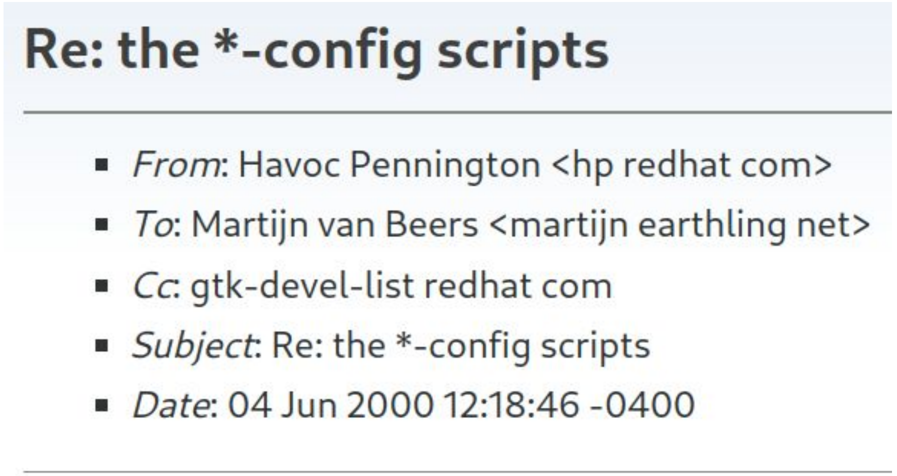
CMake 1: Primordial Flag Soup

```
SOURCE_FILES(SRCS
cmake
#...
cmSourceGroup
cmakemain
)

IF (WIN32)
  SOURCE_FILES(SRCS cmDSWWriter cmDSPWriter cmMSProjectGenerator)
ELSE (WIN32)
  SOURCE_FILES(SRCS cmUnixMakefileGenerator)
ENDIF (WIN32)

ADD_EXECUTABLE(cmake SRCS)

ADD_TEST(burn cmake)

INSTALL_TARGETS(/bin cmake)
```

kitware

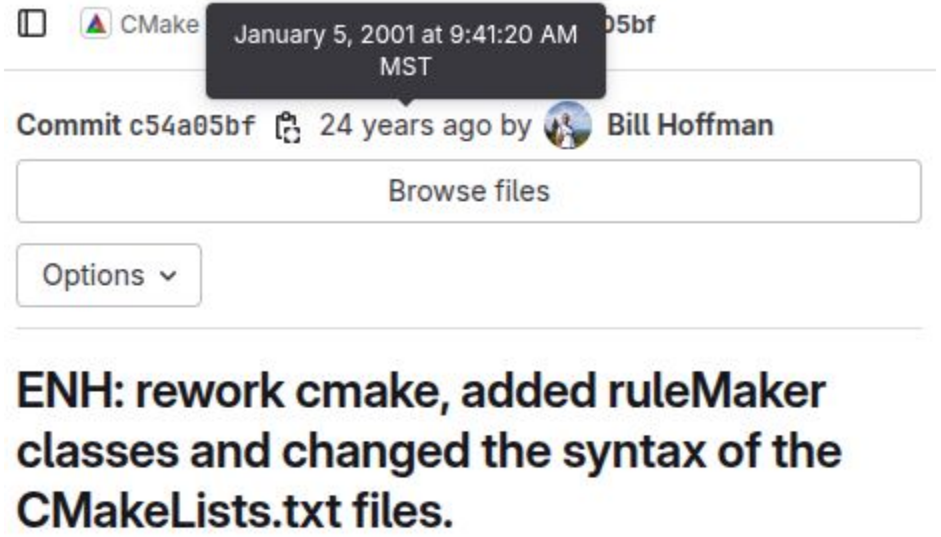
But also…

- `find_file()`
- `find_library()`
- `find_package()`
- `find_path()`
- `find_program()`



Re: the *-config scripts

- *From*: Havoc Pennington <hp redhat com>
- *To*: Martijn van Beers <martijn earthling net>
- *Cc*: gtk-devel-list redhat com
- *Subject*: Re: the *-config scripts
- *Date*: 04 Jun 2000 12:18:46 -0400



January 5, 2001 at 9:41:20 AM MST

Commit c54a05bf 24 years ago by Bill Hoffman

ENH: rework cmake, added ruleMaker classes and changed the syntax of the CMakeLists.txt files.

CMake 2: World Domination

- `add_subdirectory()`
- `cmake_policy()`
- `enable_language()`
- `export()`
- `function()`
- `install()`
- `math()`

CMake 2: World Domination

How many variables does the `project()` command set?

CMake 2: World Domination

How many variables does the `project()` command set?

PROJECT_NAME

CMake 2: World Domination

How many variables does the `project()` command set?

PROJECT_NAME
PROJECT_VERSION

CMake 2: World Domination

How many variables does the `project()` command set?

PROJECT_NAME
PROJECT_VERSION
PROJECT_BINARY_DIR
PROJECT_SOURCE_DIR

CMake 2: World Domination

How many variables does the `project()` command set?

PROJECT_NAME
PROJECT_VERSION
PROJECT_BINARY_DIR
PROJECT_SOURCE_DIR
CMAKE_PROJECT_NAME
PROJECT_IS_TOP_LEVEL
PROJECT-NAME_IS_TOP_LEVEL
CMAKE_PROJECT_VERSION
PROJECT-NAME_VERSION
CMAKE_PROJECT_VERSION_MAJOR
PROJECT_VERSION_MAJOR
PROJECT-NAME_VERSION_MAJOR
CMAKE_PROJECT_VERSION_MINOR
PROJECT_VERSION_MINOR
PROJECT-NAME_VERSION_MINOR
CMAKE_PROJECT_VERSION_PATCH
PROJECT_VERSION_PATCH
PROJECT-NAME_VERSION_PATCH
CMAKE_PROJECT_VERSION_TWEAK
PROJECT_VERSION_TWEAK
PROJECT-NAME_VERSION_TWEAK
CMAKE_PROJECT_DESCRIPTION
PROJECT_DESCRIPTION
PROJECT-NAME_DESCRIPTION
CMAKE_PROJECT_HOMEPAGE_URL
PROJECT_HOMEPAGE_URL
PROJECT-NAME_HOMEPAGE_URL
CMAKE_PROJECT_COMPAT_VERSION
PROJECT_COMPAT_VERSION
PROJECT-NAME_COMPAT_VERSION

CMake 3: I hope you like `target_*` commands

- `target_compile_definitions()`
- `target_compile_features()`
- `target_compile_options()`
- `target_include_directories()`
- `target_link_directories()`
- `target_link_options()`
- `target_precompile_headers()`
- `target_sources()`

CMake 3: I hope you like `target_*` commands

- `PRIVATE`
- `INTERFACE`
- `PUBLIC`

CMake 3: I hope you like `target_*` commands

- PRIVATE
- INTERFACE
- PUBLIC

```
target_link_libraries(VitoLib
    [PRIVATE | INTERFACE | PUBLIC]
    <lib>...)
```

CMake 3: I hope you like `target_*` commands

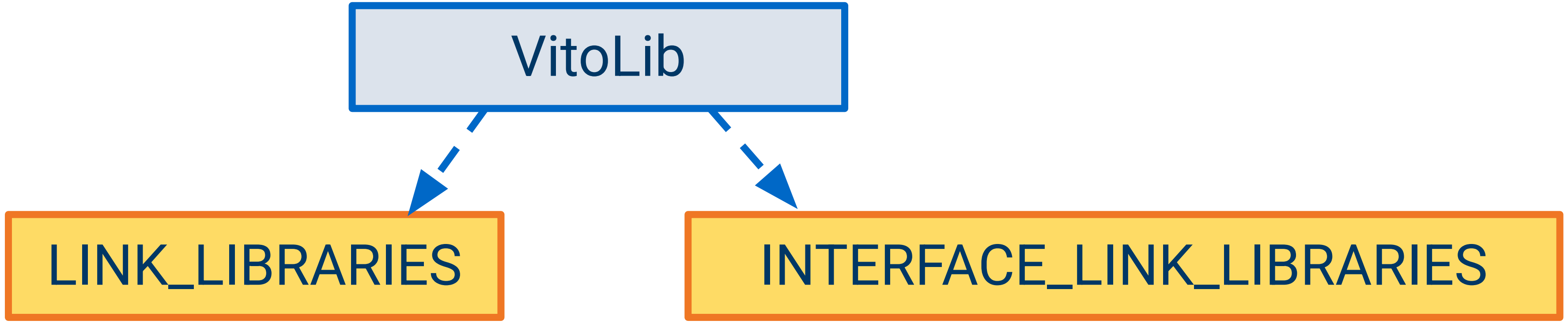
- PRIVATE
- INTERFACE
- PUBLIC

```
target_link_libraries(VitoLib
  [PRIVATE | INTERFACE | PUBLIC]
  <lib>...)
```

CMake 3: I hope you like `target_*` commands

- PRIVATE
- INTERFACE
- PUBLIC

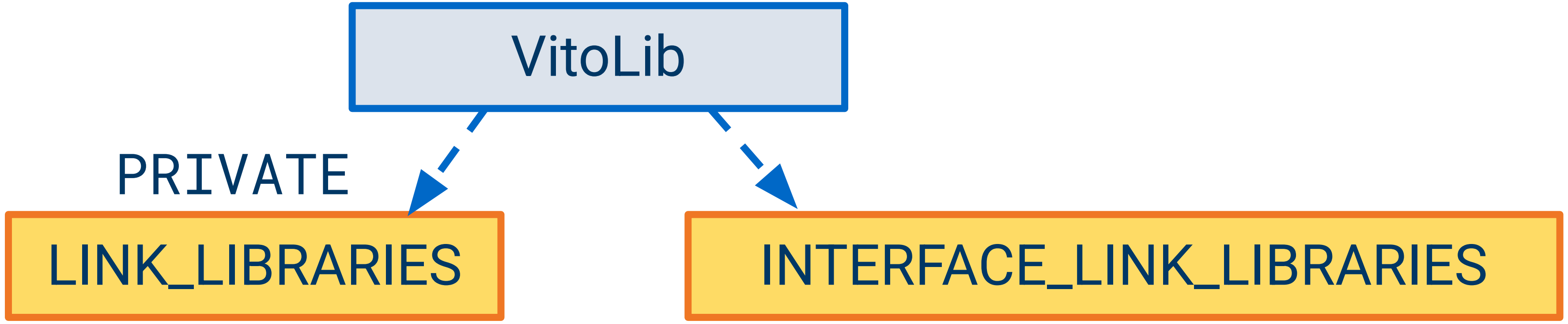
```
target_link_libraries(VitoLib
    [PRIVATE | INTERFACE | PUBLIC]
    <lib>...)
```



CMake 3: I hope you like `target_*` commands

- PRIVATE
- INTERFACE
- PUBLIC

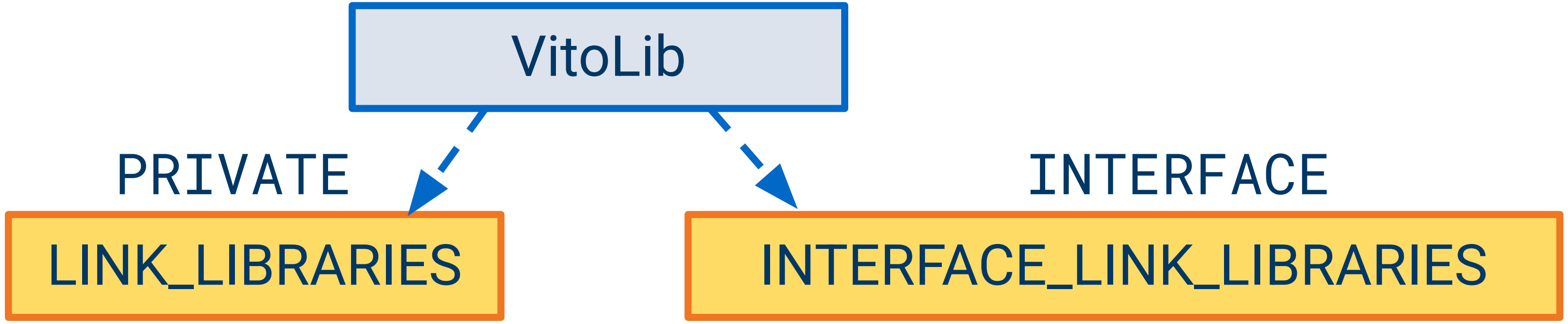
```
target_link_libraries(VitoLib
    [PRIVATE | INTERFACE | PUBLIC]
    <lib>...)
```



CMake 3: I hope you like `target_*` commands

- PRIVATE
- INTERFACE
- PUBLIC

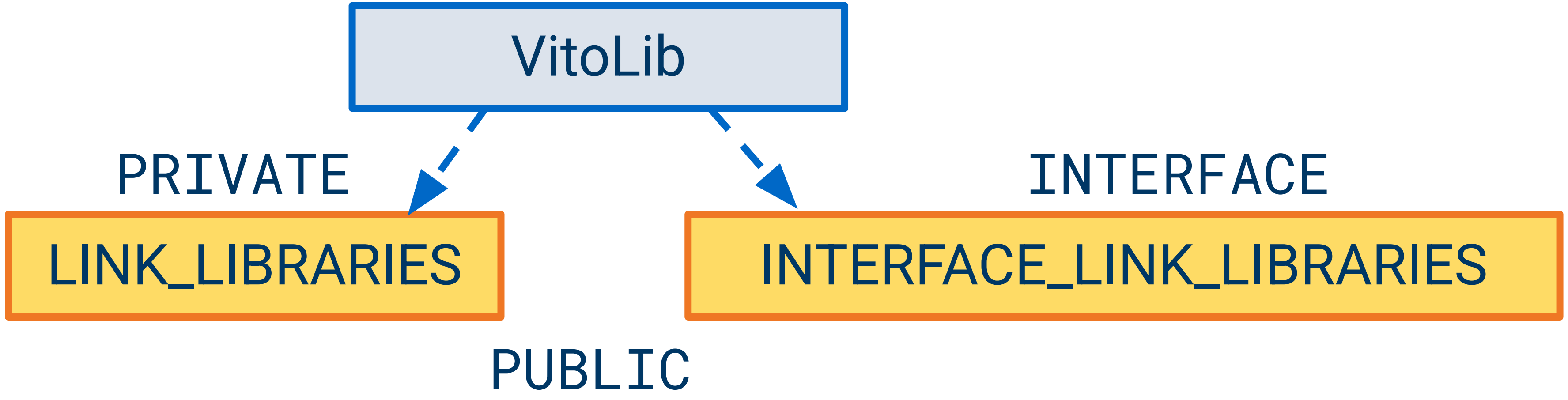
```
target_link_libraries(VitoLib
    [PRIVATE | INTERFACE | PUBLIC]
    <lib>...)
```



CMake 3: I hope you like `target_*` commands

- PRIVATE
- INTERFACE
- PUBLIC

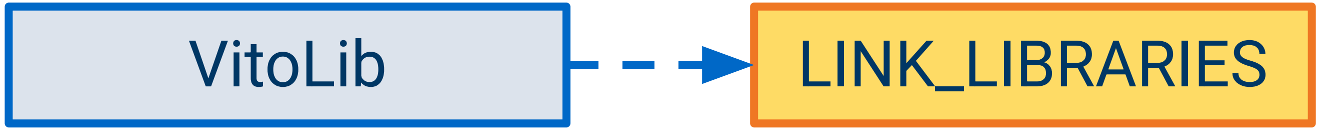
```
target_link_libraries(VitoLib
    [PRIVATE | INTERFACE | PUBLIC]
    <lib>...)
```



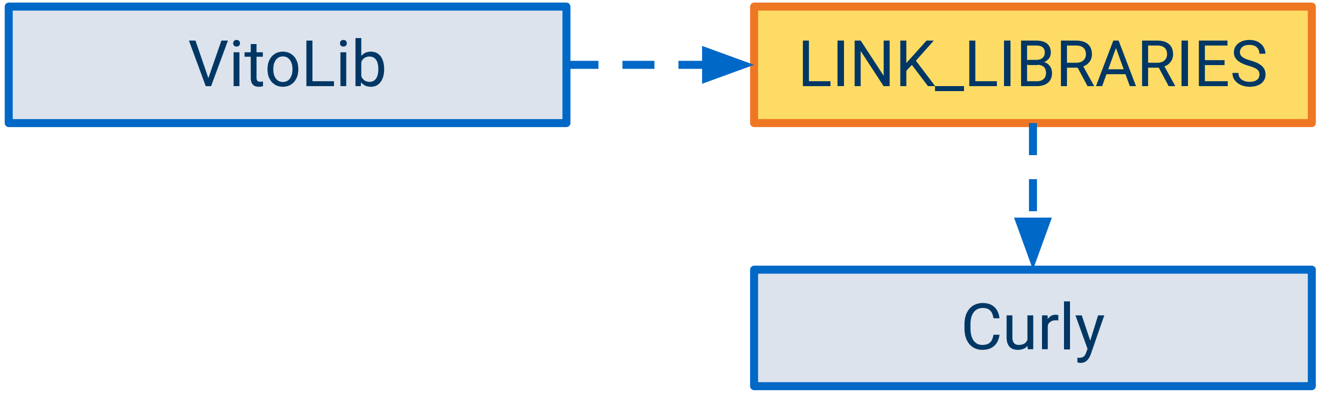
CMake 3: I hope you like `target_*` commands

VitoLib

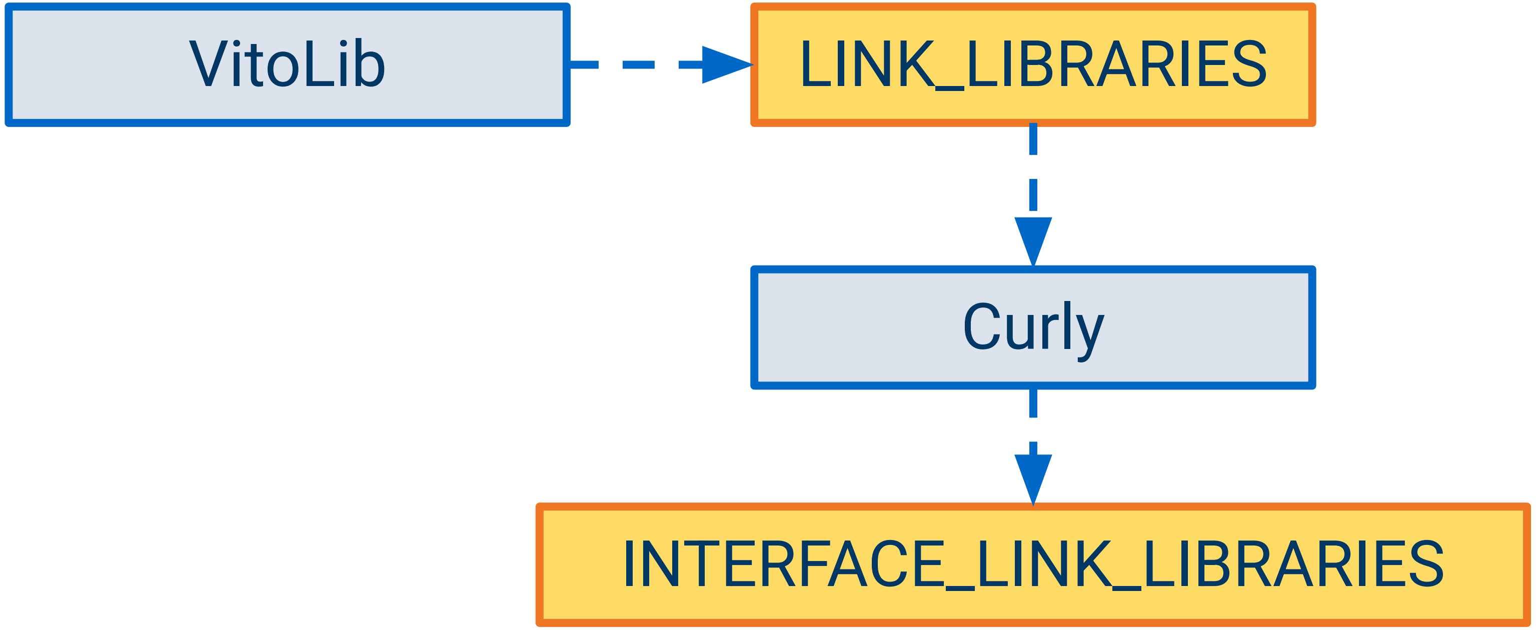
CMake 3: I hope you like `target_*` commands



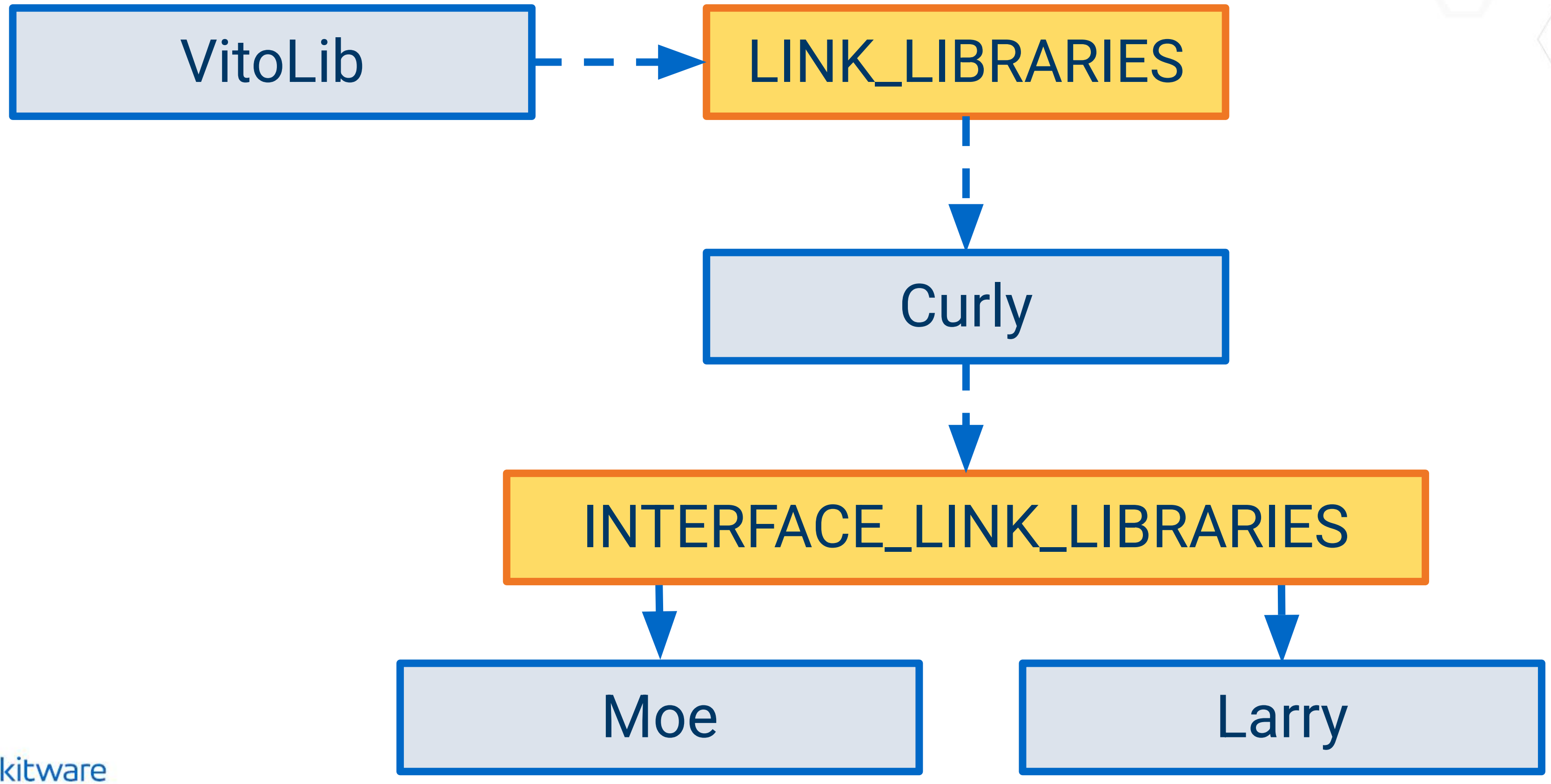
CMake 3: I hope you like `target_*` commands



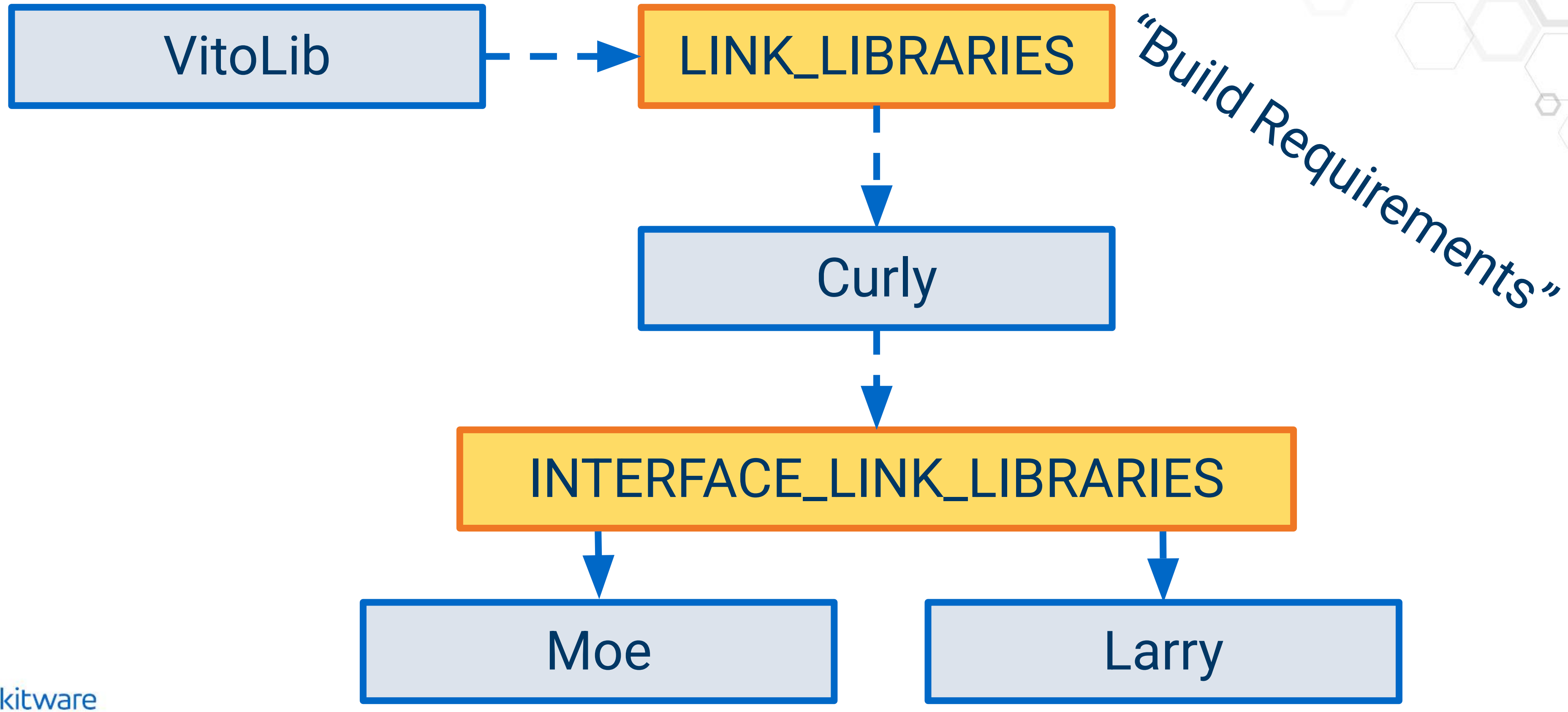
CMake 3: I hope you like `target_*` commands



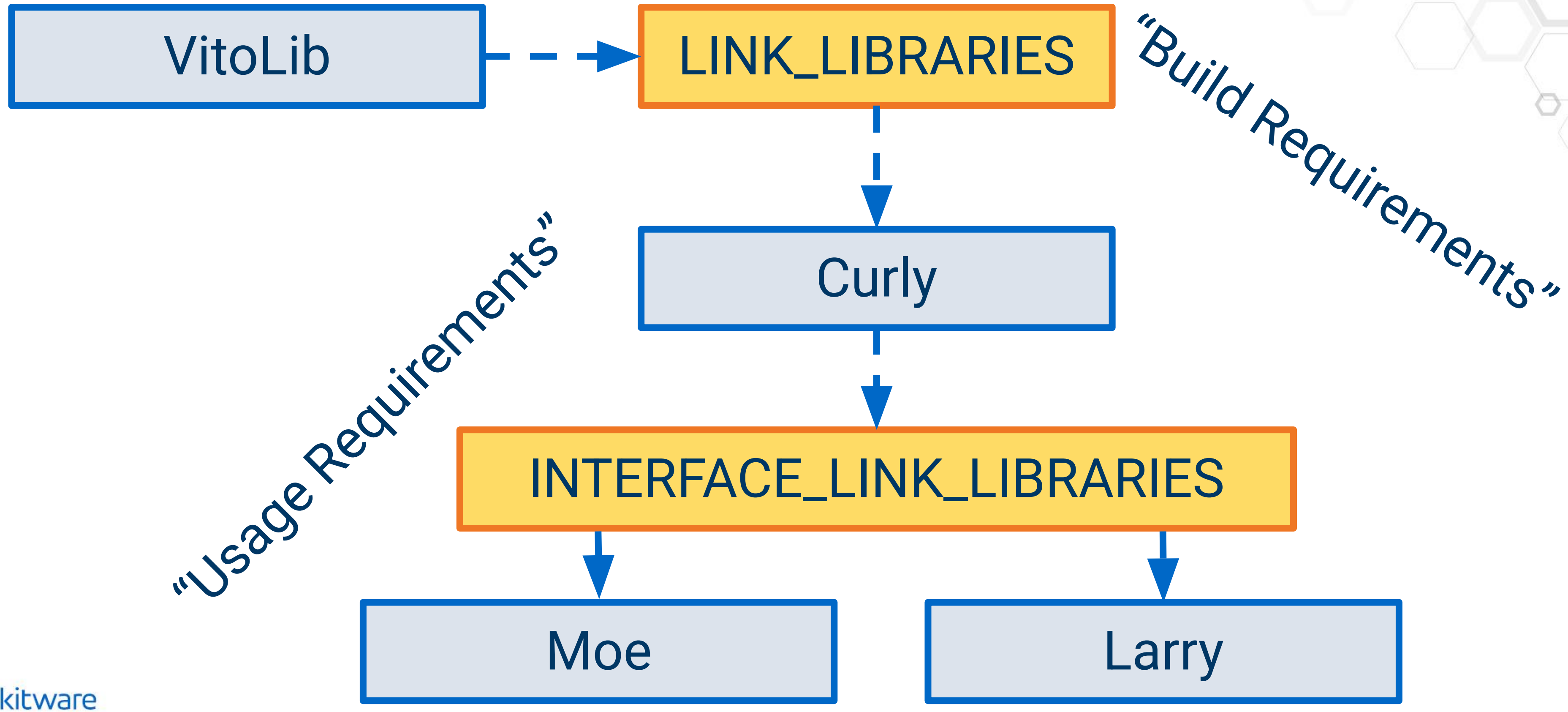
CMake 3: I hope you like `target_*` commands



CMake 3: I hope you like `target_*` commands



CMake 3: I hope you like `target_*` commands



CMake 4: ???

CMake 4: The Packaging

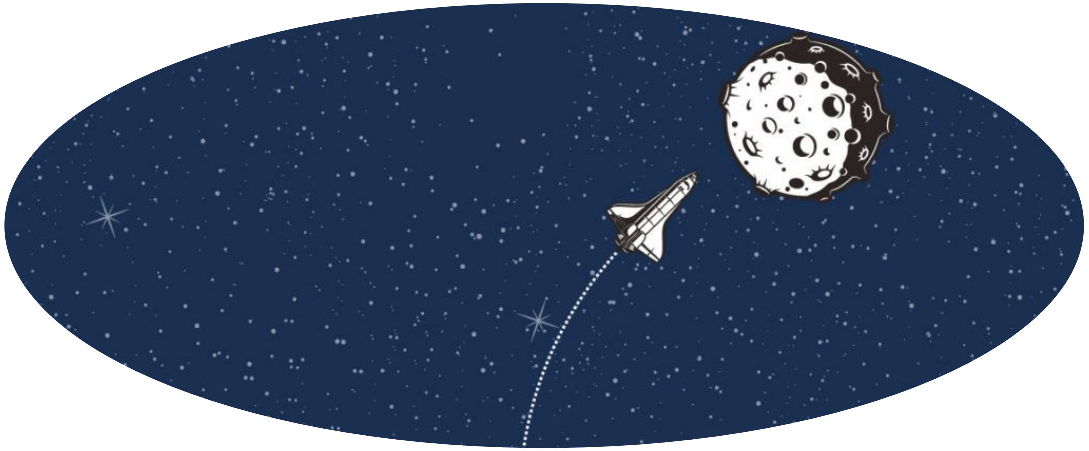
- `install()`
- `find_package()`

CMake 4: The Packaging

- `install()`
- `find_package()`

CMake 4: The Packaging

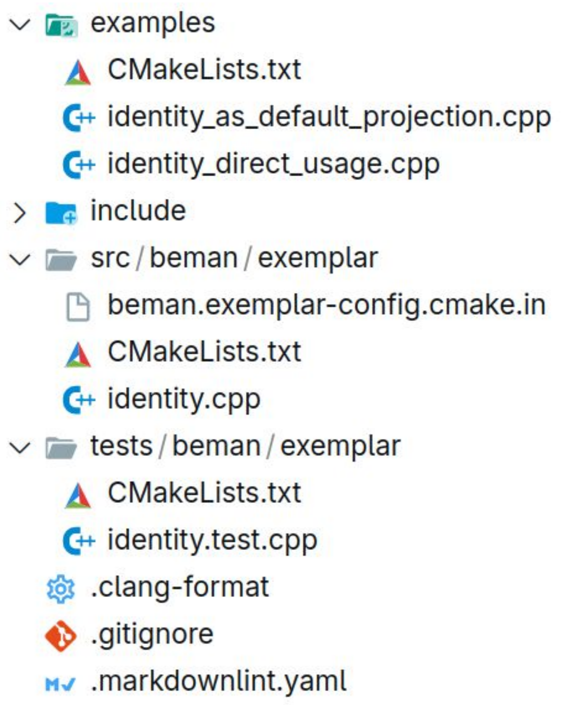
- `install()`
- `find_package()`



A Tale of Three Trees

A Tale of Three Trees

Source Tree

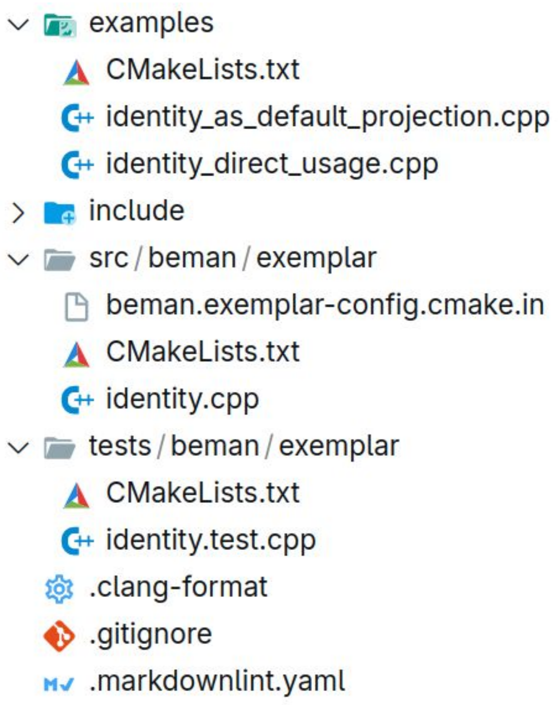


```
∨ examples
    CMakeLists.txt
    identity_as_default_projection.cpp
    identity_direct_usage.cpp
> include
∨ src / beman / exemplar
    beman.exemplar-config.cmake.in
    CMakeLists.txt
    identity.cpp
∨ tests / beman / exemplar
    CMakeLists.txt
    identity.test.cpp
  .clang-format
  .gitignore
  .markdownlint.yaml
```

kitware

A Tale of Three Trees

Source Tree



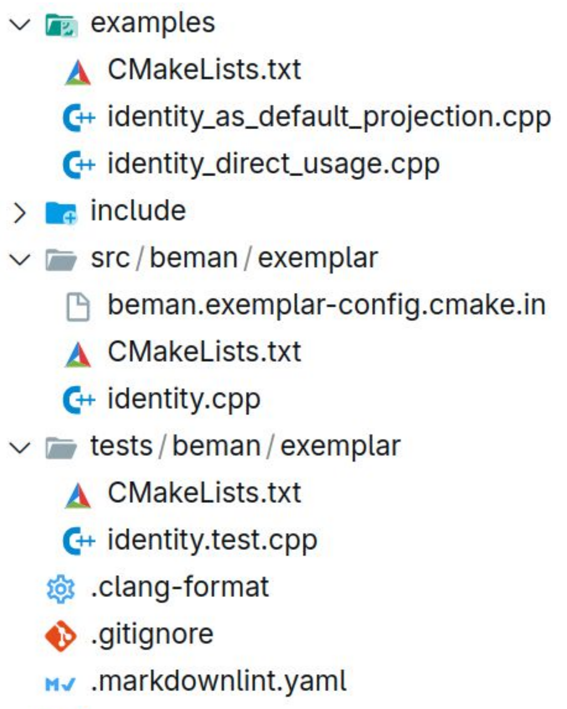
```
⌄ examples
    CMakeLists.txt
    identity_as_default_projection.cpp
    identity_direct_usage.cpp
› include
⌄ src / beman / exemplar
    beman.exemplar-config.cmake.in
    CMakeLists.txt
    identity.cpp
⌄ tests / beman / exemplar
    CMakeLists.txt
    identity.test.cpp
  .clang-format
  .gitignore
  .markdownlint.yaml
```

Keep Out

kitware

A Tale of Three Trees

Source Tree



```
∨ examples
    CMakeLists.txt
    identity_as_default_projection.cpp
    identity_direct_usage.cpp
> include
∨ src / beman / exemplar
    beman.exemplar-config.cmake.in
    CMakeLists.txt
    identity.cpp
∨ tests / beman / exemplar
    CMakeLists.txt
    identity.test.cpp
  .clang-format
  .gitignore
  .markdownlint.yaml
```

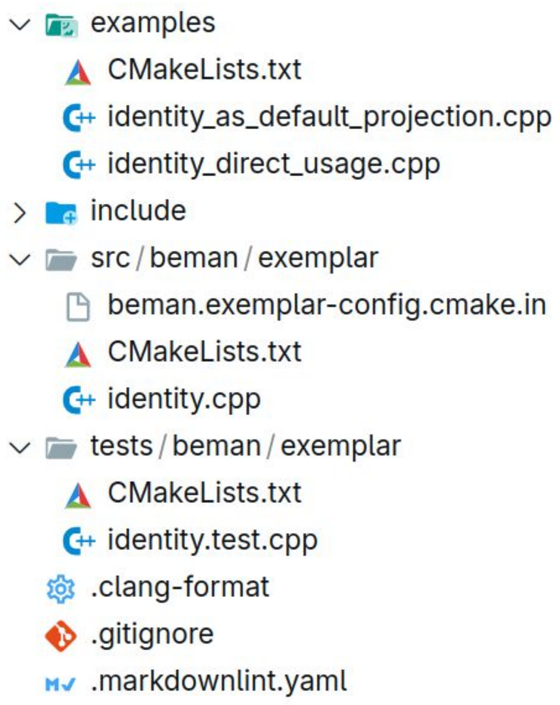
Keep Out

- Other projects’ source trees

kitware

A Tale of Three Trees

Source Tree



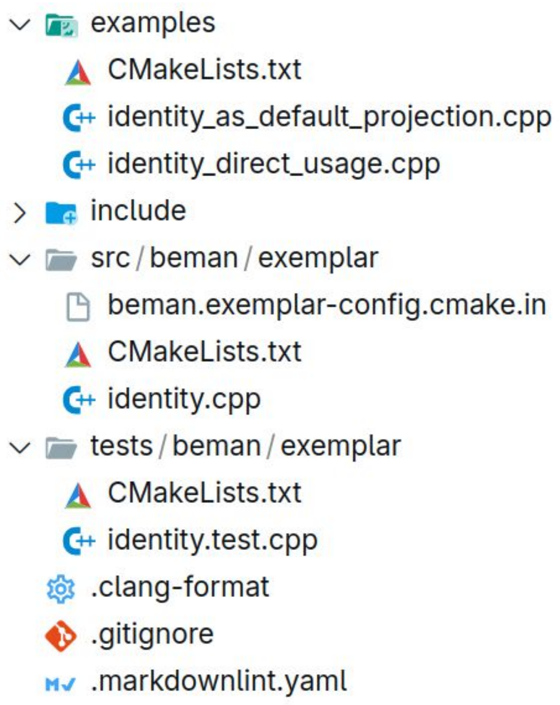
```
∨ examples
    CMakeLists.txt
    identity_as_default_projection.cpp
    identity_direct_usage.cpp
> include
∨ src / beman / exemplar
    beman.exemplar-config.cmake.in
    CMakeLists.txt
    identity.cpp
∨ tests / beman / exemplar
    CMakeLists.txt
    identity.test.cpp
  .clang-format
  .gitignore
  .markdownlint.yaml
```

Keep Out

- Other projects’ source trees
- That includes git submodules

A Tale of Three Trees

Source Tree



```
∨ examples
    CMakeLists.txt
    identity_as_default_projection.cpp
    identity_direct_usage.cpp
> include
∨ src / beman / exemplar
    beman.exemplar-config.cmake.in
    CMakeLists.txt
    identity.cpp
∨ tests / beman / exemplar
    CMakeLists.txt
    identity.test.cpp
  .clang-format
  .gitignore
  .markdownlint.yaml
```

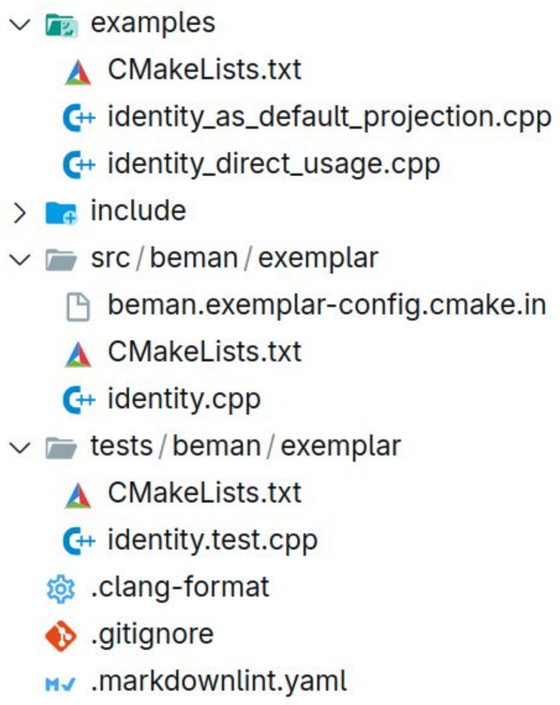
Keep Out

- Other projects' source trees
- That includes git submodules

kitware

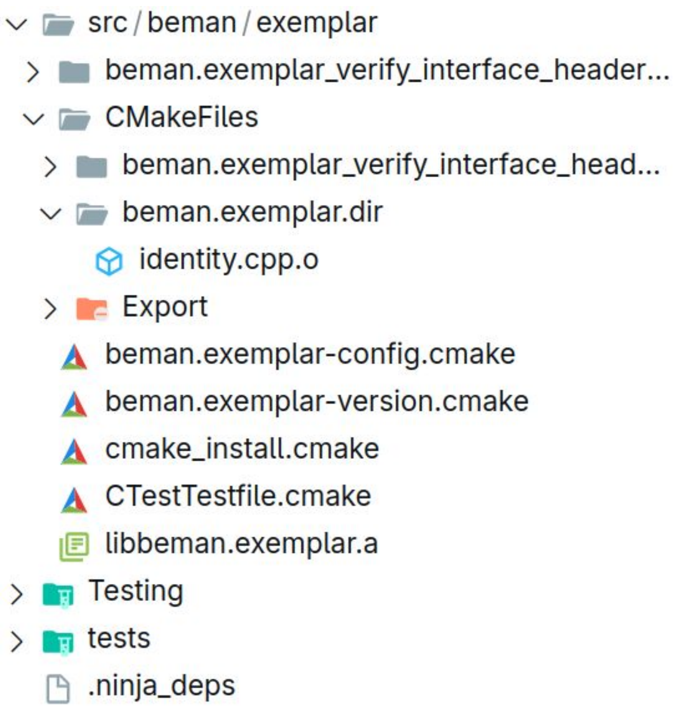
A Tale of Three Trees

Source Tree



```
⌄ examples
    CMakeLists.txt
    identity_as_default_projection.cpp
    identity_direct_usage.cpp
› include
⌄ src / beman / exemplar
    beman.exemplar-config.cmake.in
    CMakeLists.txt
    identity.cpp
⌄ tests / beman / exemplar
    CMakeLists.txt
    identity.test.cpp
  .clang-format
  .gitignore
  .markdownlint.yaml
```

Build Tree

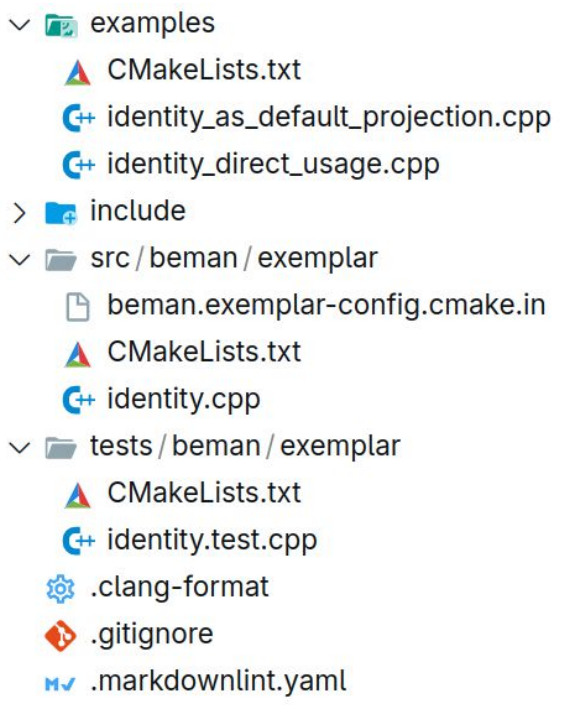


```
⌄ src / beman / exemplar
  › beman.exemplar_verify_interface_header...
  ⌄ CMakeFiles
    › beman.exemplar_verify_interface_head...
    ⌄ beman.exemplar.dir
        identity.cpp.o
    › Export
    beman.exemplar-config.cmake
    beman.exemplar-version.cmake
    cmake_install.cmake
    CTestTestfile.cmake
    libbeman.exemplar.a
› Testing
› tests
  .ninja_deps
```

kitware

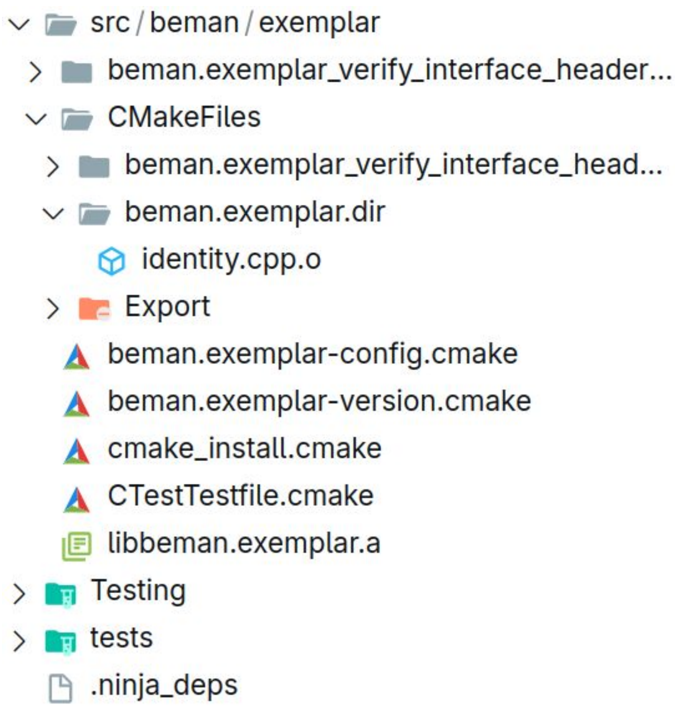
A Tale of Three Trees

Source Tree



```
examples
    CMakeLists.txt
    identity_as_default_projection.cpp
    identity_direct_usage.cpp
include
src / beman / exemplar
    beman.exemplar-config.cmake.in
    CMakeLists.txt
    identity.cpp
tests / beman / exemplar
    CMakeLists.txt
    identity.test.cpp
  .clang-format
  .gitignore
  .markdownlint.yaml
```

Build Tree



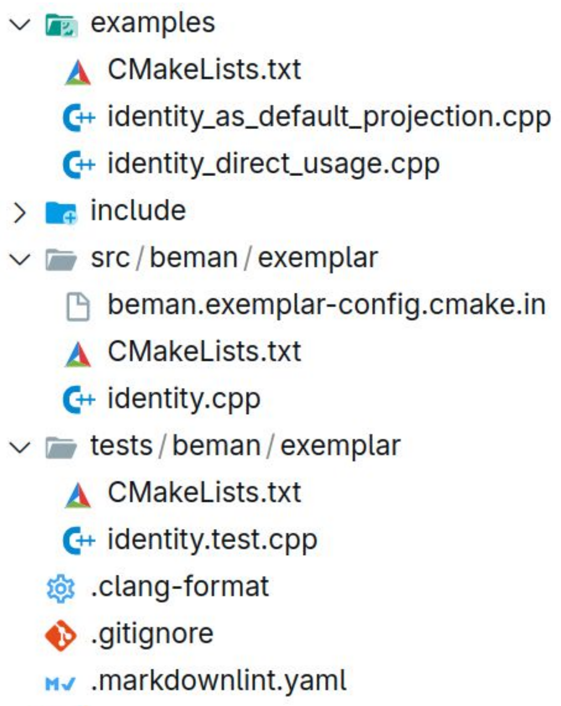
```
src / beman / exemplar
    beman.exemplar_verify_interface_header...
    CMakeFiles
        beman.exemplar_verify_interface_head...
        beman.exemplar.dir
            identity.cpp.o
        Export
    beman.exemplar-config.cmake
    beman.exemplar-version.cmake
    cmake_install.cmake
    CTestTestfile.cmake
    libbeman.exemplar.a
Testing
tests
  .ninja_deps
```

"Artifacts"

kitware

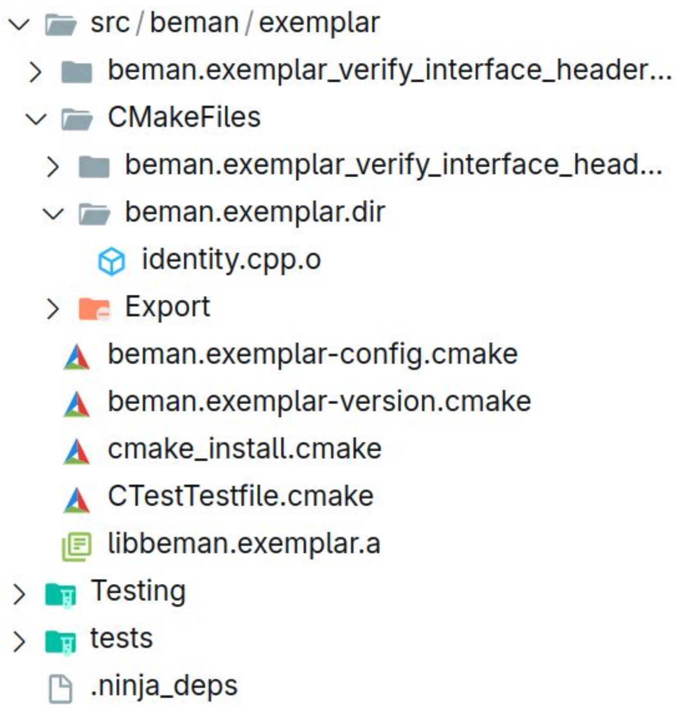
A Tale of Three Trees

Source Tree



```
∨ examples
    CMakeLists.txt
    identity_as_default_projection.cpp
    identity_direct_usage.cpp
> include
∨ src / beman / exemplar
    beman.exemplar-config.cmake.in
    CMakeLists.txt
    identity.cpp
∨ tests / beman / exemplar
    CMakeLists.txt
    identity.test.cpp
  .clang-format
  .gitignore
  .markdownlint.yaml
```

Build Tree



```
∨ src / beman / exemplar
  > beman.exemplar_verify_interface_header...
  ∨ CMakeFiles
    > beman.exemplar_verify_interface_head...
    ∨ beman.exemplar.dir
        identity.cpp.o
    > Export
    beman.exemplar-config.cmake
    beman.exemplar-version.cmake
    cmake_install.cmake
    CTestTestfile.cmake
    libbeman.exemplar.a
> Testing
> tests
  .ninja_deps
```

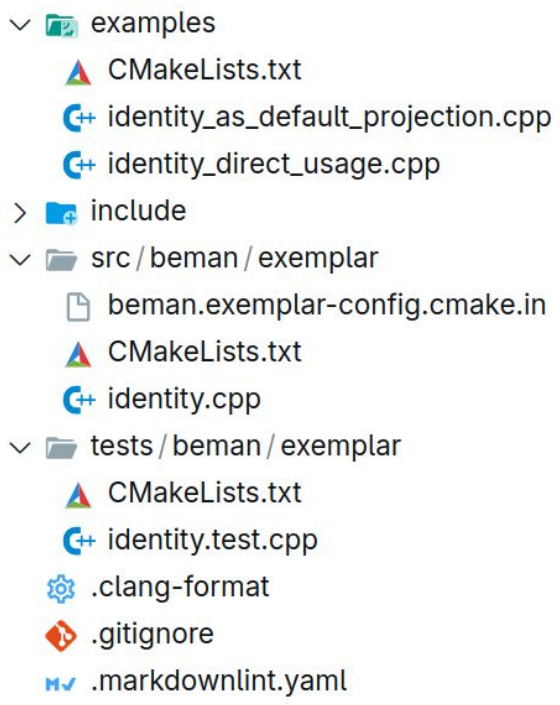
"Artifacts"

$\downarrow$

"Stuff"

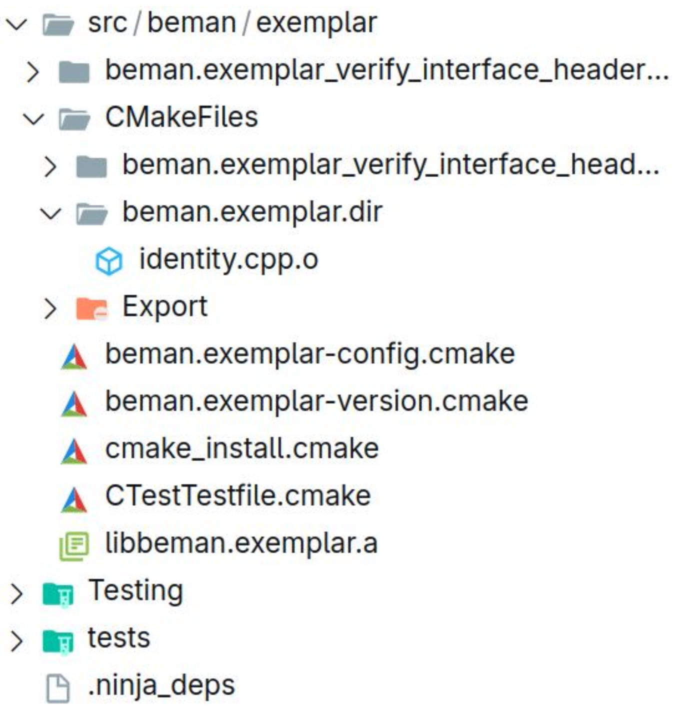
A Tale of Three Trees

Source Tree



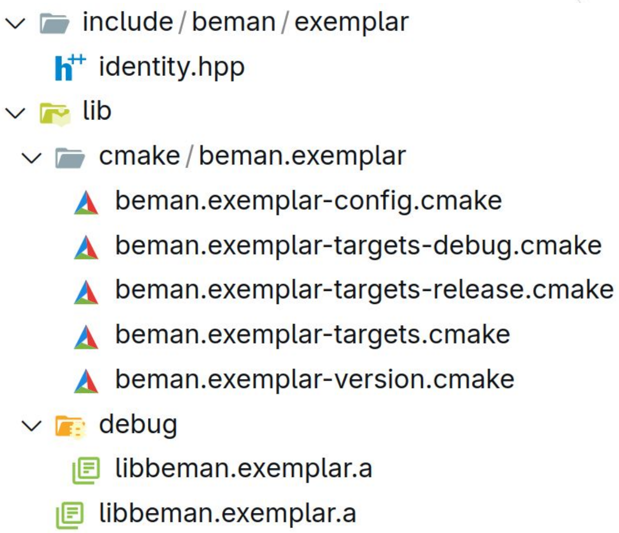
```
examples
    CMakeLists.txt
    identity_as_default_projection.cpp
    identity_direct_usage.cpp
include
src / beman / exemplar
    beman.exemplar-config.cmake.in
    CMakeLists.txt
    identity.cpp
tests / beman / exemplar
    CMakeLists.txt
    identity.test.cpp
.clang-format
.gitignore
.markdownlint.yaml
```

Build Tree



```
src / beman / exemplar
    beman.exemplar_verify_interface_header...
    CMakeFiles
        beman.exemplar_verify_interface_head...
        beman.exemplar.dir
            identity.cpp.o
        Export
    beman.exemplar-config.cmake
    beman.exemplar-version.cmake
    cmake_install.cmake
    CTestTestfile.cmake
    libbeman.exemplar.a
Testing
tests
.ninja_deps
```

Install Tree



```
include / beman / exemplar
    identity.hpp
lib
    cmake / beman.exemplar
        beman.exemplar-config.cmake
        beman.exemplar-targets-debug.cmake
        beman.exemplar-targets-release.cmake
        beman.exemplar-targets.cmake
        beman.exemplar-version.cmake
    debug
        libbeman.exemplar.a
    libbeman.exemplar.a
```

kitware

How to Describe A Source Tree In Post-Modern CMake

```
target_sources(<target>
  <INTERFACE|PUBLIC|PRIVATE>
    FILE_SET <set>
    TYPE <type>
    BASE_DIRS
      <dirs>...
    FILES
      <files>...
)
```

```
target_sources(VitoLib
  PRIVATE
    FILE_SET privateHeaders
    TYPE HEADERS
    BASE_DIRS
      ${CMAKE_CURRENT_SOURCE_DIR}
    FILES
      vito.hpp
      rishyak.hpp
)
```

kitware

How to Describe A Source Tree In Post-Modern CMake

```
target_sources(<target>
  <INTERFACE|PUBLIC|PRIVATE>
    FILE_SET <set>
    TYPE <type>
    BASE_DIRS
      <dirs>...
    FILES
      <files>...
)
```

```
target_sources(VitoLib
  PRIVATE
    FILE_SET privateHeaders
    TYPE HEADERS
    BASE_DIRS
      ${CMAKE_CURRENT_SOURCE_DIR}
      ${CMAKE_PROJECT_DIR}/include
    FILES
      vito.hpp
      ${CMAKE_CURRENT_SOURCE_DIR}/rishyak.hpp
      ${CMAKE_PROJECT_DIR}/include/brett.hpp
)
```

How to Describe A Source Tree In Post-Modern CMake

```
target_sources(VitoLib
  PRIVATE
    FILE_SET privateHeaders
    TYPE HEADERS
    BASE_DIRS
      ${CMAKE_CURRENT_SOURCE_DIR}
    FILES
      vito.hpp
      rishyak.hpp
)
```

```
target_sources(VitoLib
  PRIVATE
    FILE_SET privateHeaders
    TYPE HEADERS
    BASE_DIRS
      ${CMAKE_CURRENT_SOURCE_DIR}
    FILES
      vito.hpp
      rishyak.hpp
)
```

How to Describe A Source Tree In Post-Modern CMake

```
target_sources(VitoLib
  PRIVATE
    FILE_SET privateHeaders
    TYPE HEADERS
    BASE_DIRS
      ${CMAKE_CURRENT_SOURCE_DIR}
    FILES
      vito.hpp
      rishyak.hpp
)
```

```
target_sources(VitoLib
  PRIVATE
    FILE_SET HEADERS
    BASE_DIRS
      ${CMAKE_CURRENT_SOURCE_DIR}
    FILES
      vito.hpp
      rishyak.hpp
)
```

How to Describe A Source Tree In Post-Modern CMake

```
target_sources(VitoLib
  PRIVATE
    FILE_SET privateHeaders
    TYPE HEADERS
    BASE_DIRS
      ${CMAKE_CURRENT_SOURCE_DIR}
    FILES
      vito.hpp
      rishyak.hpp
)
```

```
target_sources(VitoLib
  PRIVATE
    FILE_SET HEADERS
    FILES
      vito.hpp
      rishyak.hpp
)
```

kitware

How to Describe A Source Tree In Post-Modern CMake

```cmake
target_sources(VitoLib
  PRIVATE
    FILE_SET privateHeaders
    TYPE HEADERS
    BASE_DIRS
      ${CMAKE_CURRENT_SOURCE_DIR}
    FILES
      vito.hpp
      rishyak.hpp
)
```

```cmake
target_sources(VitoLib
  PRIVATE
    FILE_SET HEADERS
)
```

How to Describe A Source Tree In Post-Modern CMake

```
target_sources(VitoLib
  PRIVATE
    FILE_SET privateHeaders
    TYPE HEADERS
    BASE_DIRS
      ${CMAKE_CURRENT_SOURCE_DIR}
    FILES
      vito.hpp
      rishyak.hpp
)
```

“Post-Modern”

```
target_sources(VitoLib
  PRIVATE
    FILE_SET HEADERS
)
```

kitware

How to Describe A Source Tree In Post-Modern CMake

```
target_sources(VitoLib
  PRIVATE
    FILE_SET privateHeaders
    TYPE HEADERS
    BASE_DIRS
      ${CMAKE_CURRENT_SOURCE_DIR}
    FILES
      vito.hpp
      rishyak.hpp
)
```

“Post-Modern”

```
target_sources(VitoLib
  PRIVATE
    FILE_SET HEADERS
)
```

“Modern”

```
target_include_directories(VitoLib
  PRIVATE
    ${CMAKE_CURRENT_SOURCE_DIR}
)
```

How to Describe A Source Tree In Post-Modern CMake

```
target_sources(VitoLib
  PRIVATE
    FILE_SET HEADERS
)
```

How to Describe A Source Tree In Post-Modern CMake

```
target_sources(VitoLib
  PRIVATE
    FILE_SET HEADERS
)
```

```
target_sources(VitoLib
  PRIVATE
    FILE_SET EMBED
)
```

```
#embed <data.bin>
```

How to Describe A Source Tree In Post-Modern CMake

```
target_sources(VitoLib
  PRIVATE
    FILE_SET HEADERS
)
```

```
target_sources(VitoLib
  PRIVATE
    FILE_SET SOURCES
)
```

```
target_sources(VitoLib
  PRIVATE
    FILE_SET EMBED
)
```

```
target_sources(VitoLib
  PRIVATE
    FILE_SET CXX_MODULES
)
```

kitware

How to Describe A Source Tree In Post-Modern CMake

```
target_sources(VitoLib
  PRIVATE
    FILE_SET HEADERS
)
```

```
target_sources(VitoLib
  PRIVATE
    FILE_SET SOURCES
)
```

```
target_sources(VitoLib
  PRIVATE
    FILE_SET EMBED
)
```

```
target_sources(VitoLib
  PRIVATE
    FILE_SET CXX_MODULES
)
```

How to Describe A Source Tree In Post-Modern CMake

```cmake
target_sources(VitoLib
  PRIVATE
    FILE_SET HEADERS
)
```

https://gitlab.kitware.com/cmake/cmake/-/merge_requests/8904

```cmake
target_sources(VitoLib
  PRIVATE
    FILE_SET SOURCES
)
```

```cmake
target_sources(VitoLib
  PRIVATE
    FILE_SET EMBED
)
```

https://gitlab.kitware.com/cmake/cmake/-/issues/26584

```cmake
target_sources(VitoLib
  PRIVATE
    FILE_SET CXX_MODULES
)
```

How to Describe A Source Tree In Post-Modern CMake

```cmake
target_sources(VitoLib
  PRIVATE
    FILE_SET HEADERS
)

target_sources(VitoLib
  PRIVATE
    FILE_SET CXX_MODULES
)
```

```cmake
target_sources(VitoLib
  PRIVATE
    vito.cpp
    rishyak.cpp
    #...
)
```

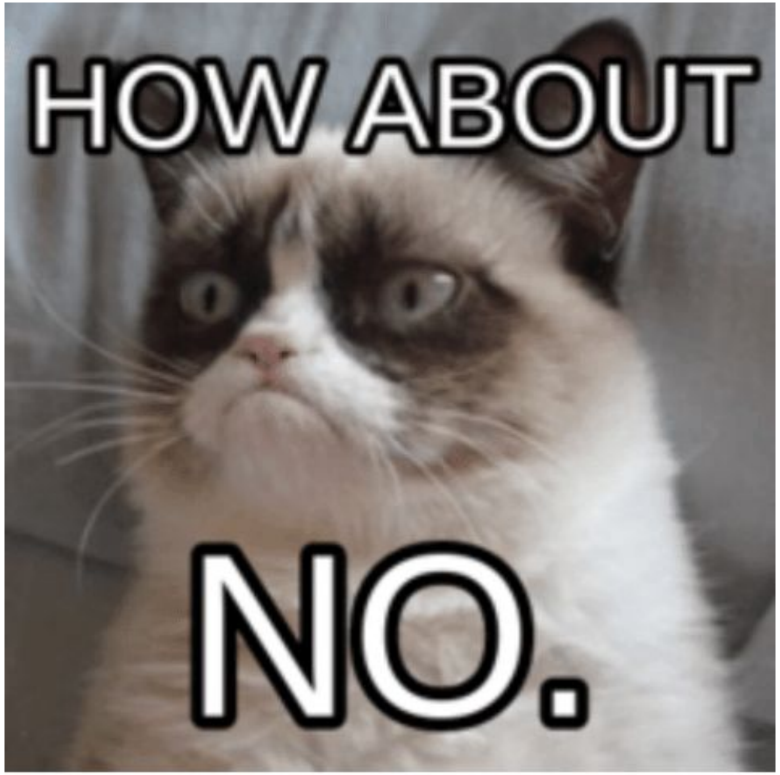
How to Describe A Source Tree In Post-Modern CMake

```
target_sources(VitoLib
  PRIVATE
    vito.cpp
    rishyak.cpp

  PRIVATE
    FILE_SET privateHeaders
    TYPE HEADERS
    BASE_DIRS
      include/internal
  ...
```

```
  ...
  PUBLIC
    FILE_SET HEADERS
    BASE_DIRS
      include
    FILES
      vito.hpp
      rishyak.hpp
)
```

How to Describe A Build Tree In Post-Modern CMake

How to Describe A Build Tree In Post-Modern CMake



How to Describe An Install Tree In Post-Modern CMake

How to Describe An Install Tree In Post-Modern CMake

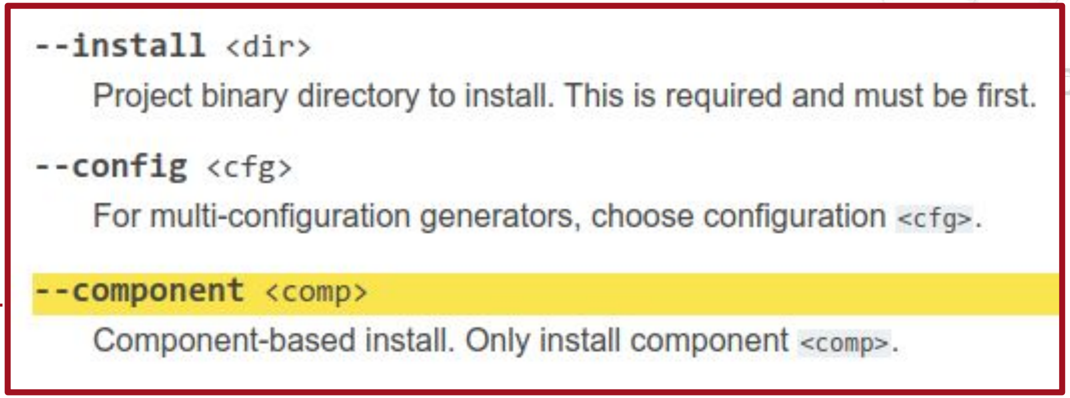
```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
  FILE_SET HEADERS
    DESTINATION ${CMAKE_INSTALL_INCLUDEDIR}
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
  FILE_SET HEADERS
    DESTINATION ${CMAKE_INSTALL_INCLUDEDIR}
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar
  DESTINATION ${CMAKE_INSTALL
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
  FILE_SET HEADERS
    DESTINATION ${CMAKE_INSTALL_INCLUDEDIR}
)
```



--install `<dir>`

Project binary directory to install. This is required and must be first.

--config `<cfg>`

For multi-configuration generators, choose configuration `<cfg>`.

--component `<comp>`

Component-based install. Only install component `<comp>`.

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
  FILE_SET HEADERS
    DESTINATION ${CMAKE_INSTALL_INCLUDEDIR}
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar-targets
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
  FILE_SET HEADERS
    DESTINATION ${CMAKE_INSTALL_INCLUDEDIR}
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.curly beman.larry beman.moe
  COMPONENT beman.exemplar
  EXPORT beman.exemplar-targets
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
  FILE_SET HEADERS
    DESTINATION ${CMAKE_INSTALL_INCLUDEDIR}
)
```

How to Describe An Install Tree In Post-Modern CMake

```cmake
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
  FILE_SET HEADERS
    DESTINATION ${CMAKE_INSTALL_INCLUDEDIR}
)
```

How to Describe An Install Tree In Post-Modern CMake

Target Type	GNUInstallDirs Variable	Built-In Default
`RUNTIME`	`${CMAKE_INSTALL_BINDIR}`	`bin`
`LIBRARY`	`${CMAKE_INSTALL_LIBDIR}`	`lib`
`ARCHIVE`	`${CMAKE_INSTALL_LIBDIR}`	`lib`
`PRIVATE_HEADER`	`${CMAKE_INSTALL_INCLUDEDIR}`	`include`
`PUBLIC_HEADER`	`${CMAKE_INSTALL_INCLUDEDIR}`	`include`
`FILE_SET` (type `HEADERS`)	`${CMAKE_INSTALL_INCLUDEDIR}`	`include`

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
  FILE_SET HEADERS
    DESTINATION ${CMAKE_INSTALL_INCLUDEDIR}
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
)
```

```
cmake \
  -DCMAKE_BUILD_TYPE=VitoRelease
  -DCMAKE_CXX_FLAGS="-O3"
```

```
${PREFIX}/lib/libbeman.exemplar.a
```

kitware

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
)
```

```
cmake \
  -DCMAKE_BUILD_TYPE=VitoDebug
  -DCMAKE_CXX_FLAGS="-ggdb -O0 -Wall -Werror"
```

```
${PREFIX}/lib/libbeman.exemplar.a
```

kitware

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exempl
  EXPORT beman.exemplar
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
)
```

```
cmake \
  -DCMAKE_BUILD_TYPE=VitoDebug \
  -DCMAKE_CXX_FLAGS="-ggdb -O0 -Wall -Werror” \
  -DCMAKE_INSTALL_LIBDIR=lib/debug
```

```
${PREFIX}/lib/debug/libbeman.exemplar.a
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exempl
  EXPORT beman.exemplar
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
)
```

```
cmake \
  -DCMAKE_BUILD_TYPE=Debug \
  -DCMAKE_CXX_FLAGS="-ggdb -O0 -Wall -Werror” \
  -DCMAKE_INSTALL_LIBDIR=lib/debug
```

```
${PREFIX}/lib/debug/debug/libbeman.exemplar.a
```

Sidebar

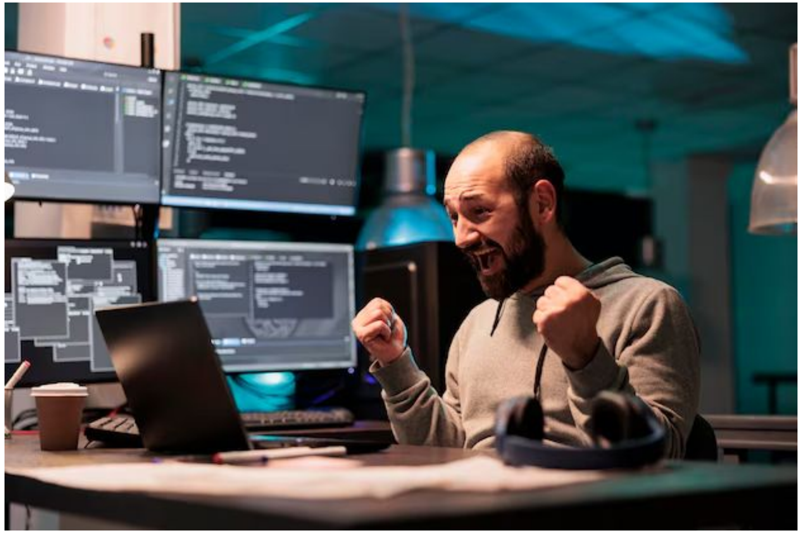
kitware

Sidebar

Mind Your Business

Sidebar

Developer

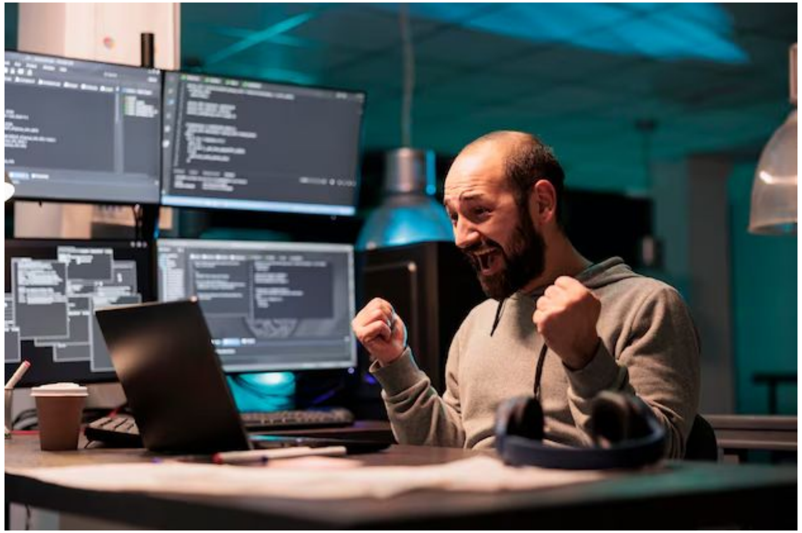


Packager

Sidebar

```
cmake \
  -DCMAKE_BUILD_TYPE=Debug \
  -DCMAKE_CXX_FLAGS="-ggdb -O0” \
  -DCMAKE_INSTALL_LIBDIR=lib/debug
```

Developer



Packager

Sidebar

Things that aren't any of your business:

- `CMAKE_*`
- `$ENV{*}`
- `install(DESTINATION)`
- Anything for which there already exists a reasonable default and is configurable by the packager

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar
  DESTINATION ${CMAKE_INSTALL_LIBDIR}$<$<CONFIG:Debug>:/debug>
  RUNTIME
    DESTINATION ${CMAKE_INSTALL_BINDIR}$<$<CONFIG:Debug>:/debug>
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  TARGETS beman.exemplar
  COMPONENT beman.exemplar
  EXPORT beman.exemplar
)

install(
  TARGETS beman.exemplar
  EXPORT beman.exemplar-targets
)
```

How to Describe An Install Tree In Post-Modern CMake

```
include(CMakePackageConfigHelpers)
configure_package_config_file(
  "${CMAKE_CURRENT_SOURCE_DIR}/beman.exemplar-config.cmake.in"
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-config.cmake"
  INSTALL_DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  PATH_VARS PROJECT_NAME PROJECT_VERSION
)
write_basic_package_version_file(
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-version.cmake"
  VERSION ${PROJECT_VERSION}
  COMPATIBILITY ExactVersion
)
```

How to Describe An Install Tree In Post-Modern CMake

```
include(CMakePackageConfigHelpers)
configure_package_config_file(
  "${CMAKE_CURRENT_SOURCE_DIR}/beman.exemplar-config.cmake.in"
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-config.cmake"
  INSTALL_DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  PATH_VARS PROJECT_NAME PROJECT_VERSION
)
write_basic_package_version_file(
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-version.cmake"
  VERSION ${PROJECT_VERSION}
  COMPATIBILITY ExactVersion
)
```

How to Describe An Install Tree In Post-Modern CMake

```
include(CMakePackageConfigHelpers)
configure_package_config_file(
  "${CMAKE_CURRENT_SOURCE_DIR}/beman.exemplar-config.cmake.in"
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-config.cmake"
  INSTALL_DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  PATH_VARS PROJECT_NAME PROJECT_VERSION
)
write_basic_package_version_file(
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-version.cmake"
  VERSION ${PROJECT_VERSION}
  COMPATIBILITY ExactVersion
)
```

How to Describe An Install Tree In Post-Modern CMake

```
include(CMakePackageConfigHelpers)
configure_package_config_file(
  "${CMAKE_CURRENT_SOURCE_DIR}/beman.exemplar-config.cmake.in"
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-config.cmake"
  INSTALL_DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  PATH_VARS PROJECT_NAME PROJECT_VERSION
)
write_basic_package_version_file(
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-version.cmake"
  VERSION ${PROJECT_VERSION}
  COMPATIBILITY ExactVersion
)
```

kitware

How to Describe An Install Tree In Post-Modern CMake

```
include(CMakePackageConfigHelpers)
configure_package_config_file(
  "${CMAKE_CURRENT_SOURCE_DIR}/beman.exemplar-config.cmake.in"
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-config.cmake"
  INSTALL_DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  PATH_VARS PROJECT_NAME PROJECT_VERSION
)
write_basic_package_version_file(
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-version.cmake"
  VERSION ${PROJECT_VERSION}
  COMPATIBILITY ExactVersion
)
```

kitware

How to Describe An Install Tree In Post-Modern CMake

```
include(CMakePackageConfigHelpers)
configure_package_config_file(
  "${CMAKE_CURRENT_SOURCE_DIR}/beman.exemplar-config.cmake.in"
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-config.cmake"
  INSTALL_DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
)

write_basic_package_version_file(
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-version.cmake"
  VERSION ${PROJECT_VERSION}
  COMPATIBILITY ExactVersion
)
```

How to Describe An Install Tree In Post-Modern CMake

```
include(CMakePackageConfigHelpers)
configure_package_config_file(
  "${CMAKE_CURRENT_SOURCE_DIR}/beman.exemplar-config.cmake.in"
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-config.cmake"
  INSTALL_DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
)

write_basic_package_version_file(
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-version.cmake"
  VERSION ${PROJECT_VERSION}
  COMPATIBILITY ExactVersion
)
```

kitware

How to Describe An Install Tree In Post-Modern CMake

```
include(CMakePackageConfigHelpers)
configure_package_config_file(
  "${CMAKE_CURRENT_SOURCE_DIR}/beman.exemplar-config.cmake.in"
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-config.cmake"
  INSTALL_DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
)

write_basic_package_version_file(
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-version.cmake"
  VERSION ${PROJECT_VERSION}
  COMPATIBILITY ExactVersion
)
```

How to Describe An Install Tree In Post-Modern CMake

```
include(CMakePackageConfigHelpers)
configure_package_config_file(
  "${CMAKE_CURRENT_SOURCE_DIR}/beman.exemplar-config.cmake.in"
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-config.cmake"
  INSTALL_DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
)

write_basic_package_version_file(
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-version.cmake"
  COMPATIBILITY ExactVersion
)
```

How to Describe An Install Tree In Post-Modern CMake

```
include(CMakePackageConfigHelpers)
configure_package_config_file(
  "${CMAKE_CURRENT_SOURCE_DIR}/beman.exemplar-config.cmake.in"
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-config.cmake"
  INSTALL_DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
)

write_basic_package_version_file(
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-version.cmake"
  COMPATIBILITY ExactVersion
)
```

How to Describe An Install Tree In Post-Modern CMake

```
include(CMakePackageConfigHelpers)
configure_package_config_file(
  "${CMAKE_CURRENT_SOURCE_DIR}/beman.exemplar-config.cmake.in"
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-config.cmake"
  INSTALL_DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
)

write_basic_package_version_file(
  "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-version.cmake"
  COMPATIBILITY ExactVersion
)
```

kitware

beman.exemplar-config.cmake

Timmy J Developer’s CMakeLists.txt

```
find_package(beman.exemplar)
```

beman.exemplar-config.cmake

Timmy J Developer’s CMakeLists.txt

```
find_package(beman.exemplar)
```

beman.exemplar-config.cmake

```
include(${CMAKE_CURRENT_LIST_DIR}/beman.exemplar-targets.cmake)
```

beman.exemplar-config.cmake

Timmy J Developer's CMakeLists.txt

```
find_package(beman.exemplar)
```

beman.exemplar-config.cmake

```
include(${CMAKE_CURRENT_LIST_DIR}/beman.exemplar-targets.cmake)
```

Timmy J Developer's CMakeLists.txt

```
find_package(beman.exemplar
  COMPONENTS Interpreter Development
)
```

beman.exemplar-config.cmake

beman.exemplar-config.cmake

```cmake
include(${CMAKE_CURRENT_LIST_DIR}/beman.exemplar-targets.cmake)

foreach(comp IN LISTS beman.exemplar_FIND_COMPONENTS)
 if(beman.exemplar_FIND_REQUIRED_${comp})
   set(beman.exemplar_FOUND FALSE)
   return()
 endif()
endforeach()
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  FILES
    "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-config.cmake"
    "${CMAKE_CURRENT_BINARY_DIR}/beman.exemplar-version.cmake"
  DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  COMPONENT beman.exemplar
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  EXPORT beman.exemplar
  DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  NAMESPACE beman::
  FILE beman.exemplar-targets.cmake
  COMPONENT beman.exemplar
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  EXPORT beman.exemplar
  DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  NAMESPACE beman::
  FILE beman.exemplar-targets.cmake
  COMPONENT beman.exemplar
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  EXPORT beman.exemplar-targets
  DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  NAMESPACE beman::
  FILE beman.exemplar-targets.cmake
  COMPONENT beman.exemplar
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  EXPORT beman.exemplar-targets
  DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  NAMESPACE beman::
  FILE beman.exemplar-targets.cmake
  COMPONENT beman.exemplar
)
```

kitware

How to Describe An Install Tree In Post-Modern CMake

```
install(
  EXPORT beman.exemplar-targets
  DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  NAMESPACE beman::
  FILE beman.exemplar-targets.cmake
  COMPONENT beman.exemplar
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  EXPORT beman.exemplar
  DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  NAMESPACE beman::
  FILE beman.exemplar-targets.cmake
  COMPONENT beman.exemplar
)
```

Wait what?

beman.exemplar-config.cmake

```
include(${CMAKE_CURRENT_LIST_DIR}/beman.exemplar-targets.cmake)
```

```
install(
  EXPORT beman.exemplar
  DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  NAMESPACE beman::
  FILE beman.exemplar-targets.cmake
  COMPONENT beman.exemplar
)
```

kitware

Wait what?

beman.exemplar-config.cmake

```
include(${CMAKE_CURRENT_LIST_DIR}/beman.exemplar-targets.cmake)
```

```
install(
  EXPORT beman.exemplar
  DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  NAMESPACE beman::
  FILE beman.exemplar-targets.cmake
  COMPONENT beman.exemplar
)
```

Wait what?

beman.exemplar-config.cmake

```
include(${CMAKE_CURRENT_LIST_DIR}/beman.exemplar-targets.cmake)
```

```
install(
  EXPORT beman.exemplar
  DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  NAMESPACE beman::
  FILE beman.exemplar-config.cmake
  COMPONENT beman.exemplar
)
```

Dependencies

beman.exemplar-config.cmake

```
include(CMakeFindDependencyMacro)

find_dependency(Moe)
find_dependency(Larry)
find_dependency(Curly)

include(${CMAKE_CURRENT_LIST_DIR}/beman.exemplar-targets.cmake)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  EXPORT beman.exemplar
  DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  NAMESPACE beman::
  FILE beman.exemplar-config.cmake
  COMPONENT beman.exemplar
  EXPORT_PACKAGE_DEPENDENCIES
)
```

How to Describe An Install Tree In Post-Modern CMake

```
install(
  EXPORT beman.exemplar
  DESTINATION "${CMAKE_INSTALL_LIBDIR}/cmake/beman.exemplar"
  NAMESPACE beman::
  FILE beman.exemplar-config.cmake
  COMPONENT beman.exemplar
  EXPORT_PACKAGE_DEPENDENCIES
)
```

Standard Disclaimer: Help/dev/experimental.rst

How to Describe An Install Tree In Post-Modern CMake

- Common Package Specification
- Language & Platform agnostic package discovery mechanism
- Tool agnostic (JSON!), portable beyond the CMake ecosystem
- Open source and publicly developed, learn more at: github.com/cps-org/cps

Standard Disclaimer: Help/dev/experimental.rst

kitware

How to Describe An Install Tree In Post-Modern CMake

Old

```
install(
  EXPORT Example-targets
)

install(
  FILES
    Example-config.cmake
    Example-config-version.cmake
)
```

New

```
install(
  PACKAGE_INFO Example
  EXPORT Example-targets
)
```

Same

```
find_package(Example)
```

Standard Disclaimer: Help/dev/experimental.rst

How am I doing on time?

How am I doing on time?

Let’s Change Gears

C++ is not the only programming language

Python is a C++ Ecosystem

“The most common C++ build system in our ecosystem is Python”

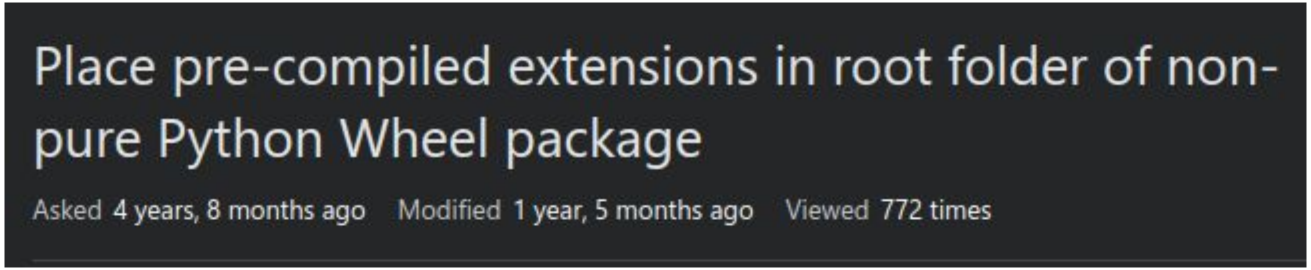
~Steve Downey

Thousands of C/C++ codebases are built, shipped, and installed every day by beginners who have never heard of a compiler.

It Used to Suck

```
from distutils.core import setup, Extension

module1 = Extension('VitoExt',
                    sources = ['vito.cpp'])

setup (name = 'PyVitoExt',
       version = '1.0',
       description = 'This is a demo package',
       ext_modules = [module1])
```

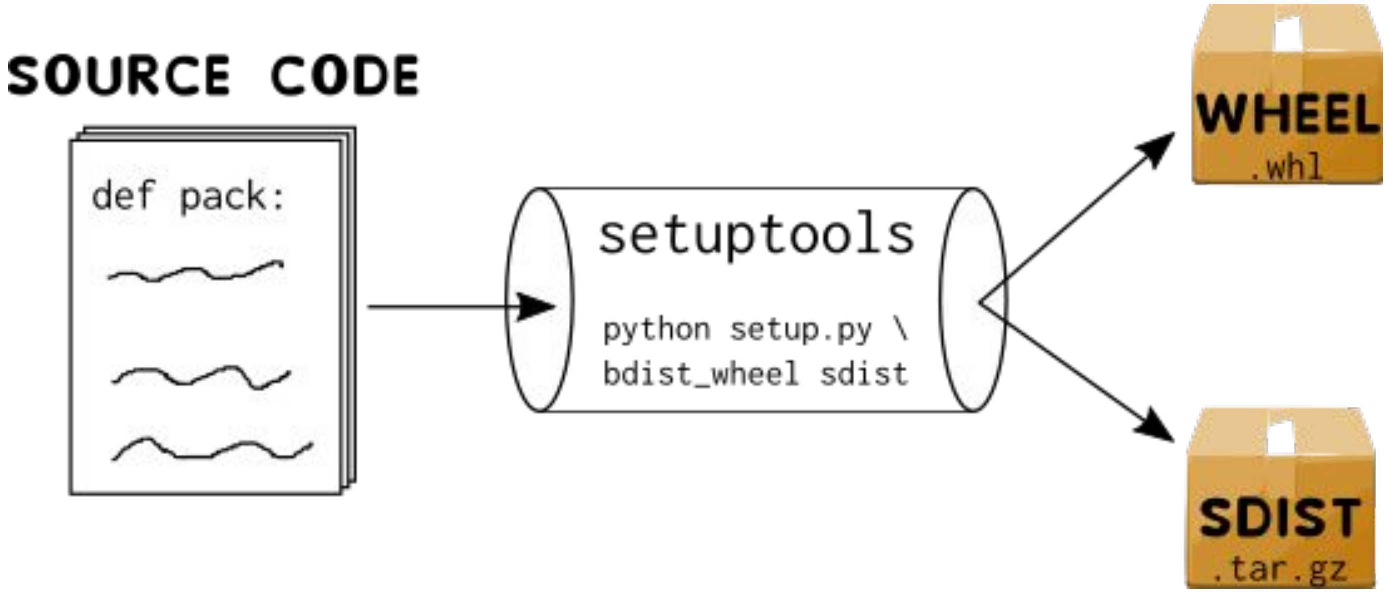
It Used to Suck


Place pre-compiled extensions in root folder of non-pure Python Wheel package

Asked 4 years, 8 months ago Modified 1 year, 5 months ago Viewed 772 times

https://stackoverflow.com/a/63436907/1201456

They Solved Packaging: PEP 427



kitware

PEP 517 & 518

PEP 517 – A build-system independent format for source trees

- A *source tree* is something like a VCS checkout
- A *build frontend* is a tool that users might run that takes arbitrary source trees or source distributions and builds them.
- The actual building is done by each source tree's *build backend*.
- An *integration frontend* is a tool that users might run that takes a set of package requirements and attempts to update a working environment to satisfy those requirements.

PEP 517 & 518

PEP 517 – A build-system independent format for source trees

- A *source tree* is something like a VCS checkout
- A *build frontend* is a tool that users might run that takes arbitrary source trees or source distributions and builds them.
- The actual building is done by each source tree's *build backend*.
- An *integration frontend* is a tool that users might run that takes a set of package requirements and attempts to update a working environment to satisfy those requirements.

PEP 517 & 518

PEP 517 – A build-system independent format for source trees

- A *source tree* is something like a VCS checkout
- A *build frontend* is a tool that users might run that takes arbitrary source trees or source distributions and builds them.
- The actual building is done by each source tree’s *build backend*.
- An *integration frontend* is a tool that users might run that takes a set of package requirements and attempts to update a working environment to satisfy those requirements.

PEP 517 & 518

PEP 517 – A build-system independent format for source trees

- A *source tree* is something like a VCS checkout
- A *build frontend* is a tool that users might run that takes arbitrary source trees or source distributions and builds them.
- The actual building is done by each source tree’s *build backend*.
- An *integration frontend* is a tool that users might run that takes a set of package requirements and attempts to update a working environment to satisfy those requirements.

pyproject.toml

```
[project]
name = "velocem"
version = "0.0.13"
description = "Hyperspeed Python Web Framework"
readme = "ReadMe.md"
requires-python = ">=3.13"
license = { "file" = "UsageLicense" }
authors = [{ "name" = "Vito Gamberini", "email" =
"vito@gamberini.email" }]
keywords = ["WSGI"]
```

pyproject.toml

```toml
dependencies = ["requests~=2.32"]

[project.optional-dependencies]
test = ["pytest", "pybench"]
```

pyproject.toml

```toml
dependencies = ["requests~=2.32"]

[project.optional-dependencies]
test = ["pytest", "pybench"]
```

PEP 751 – A file format to record Python dependencies for installation reproducibility

PEP 517 & 518

PEP 518 – Specifying Minimum Build System Requirements for Python Projects

```
[build-system]
requires = ["py-build-cmake~=0.4.0"]
build-backend = "py_build_cmake.build"
```

Python Build Backends

kitware

Let’s Change Gears

C++ and Python are not the only programming language

I’m not going to talk about Rust
... I am going to talk about Cargo

- Mostly declarative
- Sane defaults for the 99%
- Keyed off filesystem layout
- Escape hatch to Turing complete behavior

This is a CMake Talk

This is a CMake Talk

Can we fake an integration/provisioning front end?

This is a CMake Talk

Can we fake an integration/provisioning front end?

Yes, but...

We have to write it in CMakeLang

We have to write it in CMakeLang

vcpkg

CONAN

kitware

We have to write it in CMakeLang

Toolchain File

```
cmake \
  --toolchain=scripts/buildsystems/vcpkg.cmake
```

Introduction

CMake uses a toolchain of utilities to compile, link libraries and create archives, and other tasks to drive the build. The toolchain utilities available are determined by the languages enabled. In normal builds, CMake automatically determines the toolchain for host builds based on system introspection and defaults. In cross-compiling scenarios, a toolchain file may be specified with information about compiler and utility paths.

CMAKE_PROJECT_TOP_LEVEL_INCLUDES

```
cmake \
  -DCMAKE_PROJECT_TOP_LEVEL_INCLUDES=cmake/use-fetch-content.cmake
```

```
cmake_language(
  SET_DEPENDENCY_PROVIDER BemanExemplar_provideDependency
  SUPPORTED_METHODS FIND_PACKAGE
)
```

```
{"dependencies": [{
   "name": "googletest",
   "package_name": "GTest",
   "git_repository": "https://github.com/google/googletest.git",
   "git_tag": "6910c9d9165801d8827d628cb72eb7ea9dd538c5"
}]}
```

Vito's CMake Wishlist:

- CMake Formatter
- CMake Linter
- Scanner-Enabled Code Generation
- Better Package Manager Integration / Bootstrapping
- Deprecate ${MOST_RECENT_NIGHTLY_FAIL_PLATFORM}