CMake

Installing and Finding packages, Exporting and Importing targets

Alexander Neundorf
<neundorf@kde.org>

Feb 2nd 2013, FOSDEM Brussels

Outline

- find_package() MODULE mode
- find_package() CONFIG mode
- Importing targets
- Exporting targets

find_package()

```
find_package(LibXml2 2.0.0 REQUIRED)
```

- Searches a package and provides information for using it
- Cross-platform, no additional dependencies
- Provides structured, toolchain-independent information
- “classic” MODULE-mode: FindLibXml2.cmake:
  - Comes with CMake or the using application
  - Searches headers, libraries, tries to determine the version number
- CONFIG-mode: Qt5CoreConfig.cmake
  - Comes with the package
  - Searched in all standard locations
  - No guessing/testing/parsing required

find_package() MODULE mode 1/2

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)

find_package(JPEG REQUIRED)

include_directories(${JPEG_INCLUDE_DIR})
add_executable(viewer main.c)
target_link_libraries(viewer ${JPEG_LIBRARY})
```

FindJPEG.cmake (simplified):

```
find_path(JPEG_INCLUDE_DIR jpeglib.h)
find_library(JPEG_LIBRARY NAMES jpeg )

include(FindPackageHandleStandardArgs)
find_package_handle_standard_args(JPEG
                        REQUIRED_VARS  JPEG_LIBRARY JPEG_INCLUDE_DIR)
```

find_package() MODULE mode 2/2

- **`find_package`**`(JPEG)`
- Locates the Find-module (`FindJPEG.cmake`), then executes it
  - first in `${CMAKE_MODULE_PATH}`
  - then in `${CMAKE_ROOT}/Modules/`
- Results in a set of variables containing file paths
- Shipped with CMake or the application, e.g. HelloWorld
- Env.var. CMAKE_PREFIX_PATH !

find_package() CONFIG mode

- **`find_package`**`(Qt5Core NO_MODULE)`
- Searches `<Name>Config.cmake` files, e.g. `Qt5CoreConfig.cmake` and loads it
- In all usual lib/ and share/ directories
- Env.var. CMAKE_PREFIX_PATH
- Installed by the package itself
- Typically generated by CMake projects
- Contains information about the installed package:
  - Libraries, include dirs, version, enabled features, etc.
  - Typically creates imported targets
  - No testing/checking/guessing needed

CONFIG mode in detail

- **`find_package`**`(Foo 1.2.3 NO_MODULE)`
- Searches Files:
  - First `FooConfigVersion.cmake` for checking the version number
  - Then `FooConfig.cmake`
  - Alternatively `Foo-config-version.cmake` and `Foo-config.cmake`
- In the subdirs
  - **`lib/cmake/foo*/`**
  - `lib/foo*/cmake/`
  - `lib/foo*/`
  - `cmake/`
  - `foo*/`
  - `foo*/cmake/`
  - Also `share/cmake/foo*/`, `share/foo*/cmake/`, `share/foo*/`
  - Also `lib64/` and multiarch dirs
- In all usual prefixes (`/usr/local/`, `/usr/`, `/`, `CMAKE_PREFIX_PATH` etc.)

FooConfigVersion.cmake

- **find_package**(Qt5Core 5.1.0 *NO_MODULE REQUIRED*)
- For users:
  - Compares installed version with the requested version
- For developers:

```
cmake_minimum_required(VERSION 2.8.8)

include(CMakePackageConfigHelpers)
write_basic_package_version_file(FooConfigVersion.cmake
            VERSION 1.2.3
            COMPATIBILITY AnyNewerVersion|SameMajorVersion|ExactVersion)

install(FILES FooConfigVersion.cmake
        DESTINATION ${CMAKE_INSTALL_LIBDIR}/cmake/foo/)
```

FooConfig.cmake

- Simplistic example, e.g. /usr/lib/cmake/foo-1.2.3/FooConfig.cmake:

```
# Some documentation...

set(Foo_VERSION "1.2.3")

set(Foo_INCLUDE_DIRS /usr/include/foo/ )
set(Foo_LIBRARIES /usr/lib/libfoo.so)

set(Foo_SUPPORTS_Bar TRUE)
set(Foo_SUPPORTS_Blub FALSE)

include(${CMAKE_CURRENT_LIST_DIR}/FooMacros.cmake)
```

- Issues:
  - Hardcoded paths → not relocatable (especially important for Windows, OSX)
  - Debug/release builds of libraries hard to handle

Importing targets

- FooConfig.cmake:

```
...
include(${CMAKE_CURRENT_LIST_DIR}/FooTargets.cmake)
set(Foo_LIBRARIES Foo::foo )
```

- FooTargets.cmake, generated by CMake:

```
add_library(Foo::foo SHARED IMPORTED)
...
file(GLOB configFiles  "FooTargets-*.cmake")
foreach(file ${configFiles})
   include(${file})
endforeach()
```

- FooTargets-DEBUG.cmake, generated by CMake:

```
set_property(TARGET Foo::foo APPEND PROPERTY  IMPORTED_CONFIGURATIONS DEBUG)
set_target_properties(Foo::foo PROPERTIES
                      IMPORTED_LINK_INTERFACE_LIBRARIES_DEBUG "Qt5::Core"
                      IMPORTED_LOCATION_DEBUG "${_PREFIX}/lib/libfoo.so")
...
if(NOT EXISTS "${_PREFIX}/lib/libfoo.so" )
   message(FATAL_ERROR "Foo::foo references a file that does not exist.")
endif()
```

Useful Error Messages

- Since CMake 2.8.8: To help your users, always use the `MODULE` / `NO_MODULE` / `CONFIG` keywords with **`find_package()`** !
- If a Find-module is used, use **`find_package`**`(... MODULE ...)`
- If a Config.cmake file is searched directly, use **`find_package`**`(... NO_MODULE ...)`

  or **`find_package`**`(... CONFIG ...)`

New in CMake 2.8.11

- Per-target include dirs: `target_include_directories()`
- Per-target compile definitions: `target_compile_definitions()`
- For libraries: `PUBLIC`, `PRIVATE` and `INTERFACE` settings:
  - `PUBLIC`: used for building, and for using
  - `PRIVATE`: used for building only
  - `INTERFACE`: used when using only
- *Works also for imported targets !*

```
cmake_minimum_required(VERSION 2.8.11)

find_package(Foo NO_MODULE REQUIRED)

add_executable(hello main.cpp )

target_include_directories(hello PRIVATE Foo::foo /opt/blub/include )
target_compile_definitions(hello PRIVATE Foo::foo)
target_link_libraries(hello LINK_RIVATE Foo::foo)
```

Creating Config.cmake files

- Put the installed targets into an „export-set” using **install**(TARGETS ... EXPORT ...)
- Install the export-set using **install**(EXPORT ...)
- Configure a <Name>Config.cmake file using **configure_package_config_file**(...)
- Install the <Name>Config.cmake file using **install**(FILES ...)

```
add_library(foo ${srcs} )

set(CMAKECONFIG_INSTALL_DIR "${CMAKE_INSTALL_LIBDIR}/cmake/Foo")

include(CMakePackageConfigHelpers)
configure_package_config_file(FooConfig.cmake.in FooConfig.cmake
                              INSTALL_DESTINATION  ${CMAKECONFIG_INSTALL_DIR}
                              PATH_VARS  INCLUDE_INSTALL_DIR )

install(TARGETS foo  DESTINATION ${CMAKE_INSTALL_LIBDIR}  EXPORT FooTargets)

install(EXPORT FooTargets  FILE FooTargets.cmake  NAMESPACE Foo::
        DESTINATION ${CMAKECONFIG_INSTALL_DIR} )

install(FILES "${CMAKE_CURRENT_BINARY_DIR}/FooConfig.cmake"
        DESTINATION ${CMAKECONFIG_INSTALL_DIR} )
```

Writing a Config.cmake.in file

- Use **`configure_package_config_file(...)`**
- `@PACKAGE_INIT@` : replaced with code which makes the file relocatable
- `@PACKAGE_INCLUDE_INSTALL_DIR@` :
  - defined for each `PATH_VARS` variable
  - relocatable version of the input path variable
  - Input can be relative or absolute path
- **`set_and_check()`**: sets the variable and checks that the given file or path exists

```
@PACKAGE_INIT@

set(Foo_VERSION_MAJOR "@FOO_VERSION_MAJOR@")
set(Foo_VERSION_MINOR "@FOO_VERSION_MINOR@")
set(Foo_VERSION_PATCH "@FOO_VERSION_PATCH@")

set_and_check(Foo_INCLUDE_DIR    "@PACKAGE_INCLUDE_INSTALL_DIR@")

include("${CMAKE_CURRENT_LIST_DIR}/FooTargets.cmake")

set(Foo_LIBRARY Foo::foo)
```

The End

Questions ???

- **http://www.cmake.org**
- Man-Pages
- Wiki
- FAQ
- Mailinglists
- Book: "Mastering CMake"