Help! I want people to use my CMake pa

Victor Eijkhout

Fall 2023

Justification

CMake is a portable build system that is becoming a *de facto* standard for C++ package management.
By using CMake your software becomes part of the CMake eco-system.

Table of contents

Make your package discoverable

Other discovery mechanisms

Some packages come with `FindWhatever.cmake` or similar files.
Pity that there is not just one standard.
These define some macros, but you need to read the docs to see which.
Pity that there is not just one standard.
Some examples follow.

MPI

MPI from C

MPI has a module:

```
find_package( MPI )
target_include_directories(
       ${PROJECT_NAME} PUBLIC
       ${MPI_C_INCLUDE_DIRS} )
target_link_libraries(
       ${PROJECT_NAME} PUBLIC
       ${MPI_C_LIBRARIES} )
```

MPI from C++

```
find_package( MPI )
target_include_directories(
        ${PROJECT_NAME} PUBLIC
        ${MPI_CXX_INCLUDE_DIRS} )
target_link_libraries(
        ${PROJECT_NAME} PUBLIC
        ${MPI_CXX_LIBRARIES} )
```

MPI from Fortran90

```
find_package( MPI )
target_include_directories(
        ${PROJECT_NAME} PUBLIC
        ${MPI_INCLUDE_DIRS} )
target_link_directories(
        ${PROJECT_NAME} PUBLIC
        ${MPI_LIBRARY_DIRS} )
target_link_libraries(
        ${PROJECT_NAME} PUBLIC
        ${MPI_Fortran_LIBRARIES} )
```

MPI from Fortran2008

```
if( MPI_Fortran_HAVE_F08_MODULE )
else()
  message( FATAL_ERROR "No f08 module for this MPI" )
endif()
```

MPL

```
find_package( mpl REQUIRED )
target_include_directories(
        ${PROJECT_NAME} PUBLIC
        ${CMAKE_CURRENT_SOURCE_DIR}
        mpl::mpl )
target_link_libraries(
        ${PROJECT_NAME} PUBLIC
        mpl::mpl )
```

OpenMP

OpenMP from C

```
find_package(OpenMP)
target_link_libraries(
    ${PROJECT_NAME}
    PUBLIC OpenMP::OpenMP_C )
```

OpenMP from C++

```
find_package(OpenMP)
if(OpenMP_CXX_FOUND)
else()
       message( FATAL_ERROR "Could not find OpenMP" )
endif()
target_link_libraries(
    ${PROJECT_NAME}
    PUBLIC OpenMP::OpenMP_CXX )
```

OpenMP from Fortran

```
enable_language(Fortran)
find_package(OpenMP)
target_link_libraries(
    ${PROJECT_NAME}
    PUBLIC OpenMP::OpenMP_Fortran )
```

TBB

TBB

```
find_package(TBB REQUIRED)
target_link_libraries( ${PROJECT_NAME} PUBLIC TBB::tbb)
```

Other

Kokkos

```
find_package(Kokkos REQUIRED)
target_link_libraries(myTarget Kokkos::kokkos)
```

Either set `CMAKE_PREFIX_PATH` or add

```
-DKokkos_ROOT=<Kokkos Install Directory>/lib64/cmake/Kokkos
```

Maybe:

```
-DCMAKE_CXX_COMPILER=<Kokkos Install Directory>/bin/
  nvcc_wrapper
```

See https://kokkos.org/kokkos-core-wiki/ProgrammingGuide/Compiling.html

Data packages

Hdf5

C:

```
find_package( PkgConfig REQUIRED )
pkg_check_modules( NETCDF REQUIRED netcdf )

target_include_directories(
        ${PROJECTNAME} PUBLIC
        ${NETCDF_INCLUDE_DIRS} )
target_link_libraries(
        ${PROJECTNAME} PUBLIC
        ${NETCDF_LIBRARIES} )
target_link_directories(
        ${PROJECTNAME} PUBLIC
        ${NETCDF_LIBRARY_DIRS} )
target_link_libraries(
        ${PROJECTNAME} PUBLIC netcdf )
```

Hdf5, Fortran

```
find_package( PkgConfig REQUIRED )
pkg_check_modules( NETCDFF REQUIRED netcdf-fortran )
pkg_check_modules( NETCDF REQUIRED netcdf )

target_include_directories(
       ${PROJECTNAME} PUBLIC
       ${NETCDFF_INCLUDE_DIRS}
       )
target_link_libraries(
       ${PROJECTNAME} PUBLIC
       ${NETCDFF_LIBRARIES} ${NETCDF_LIBRARIES}
       )
target_link_directories(
       ${PROJECTNAME} PUBLIC
       ${NETCDFF_LIBRARY_DIRS} ${NETCDF_LIBRARY_DIRS}
       )
target_link_libraries(
       ${PROJECTNAME} PUBLIC netcdf )
```

HighFive

Third party C++ interface to hdf5

```
find_package( HighFive REQUIRED )
target_link_libraries( ${PROJECTNAME} HighFive)
```

Making your package discoverable through pkgconfig

How does pkgconfig work?

Use the `PKG_CONFIG_PATH` variable:

```
$ module show cxxopts 2>&1 | grep -i pkg
prepend_path("PKG_CONFIG_PATH","/opt/cxxopts/intel23/lib64/pkgconfig")
```

Write your own .pc file

configure_file line in CMakeLists.txt:

```
configure_file(
    ${CMAKE_CURRENT_SOURCE_DIR}/${PROJECT_NAME}.pc.in
    ${CMAKE_CURRENT_BINARY_DIR}/${PROJECT_NAME}.pc
    @ONLY)
```

Write your own `.pc` file’

The `.pc.in` file:

```
prefix="@CMAKE_INSTALL_PREFIX@"
exec_prefix="${prefix}"
libdir="${prefix}/lib"
includedir="${prefix}/include"

Name: @PROJECT_NAME@
Description: @CMAKE_PROJECT_DESCRIPTION@
Version: @PROJECT_VERSION@
Cflags: -I${includedir}
Libs: -L${libdir} -l@libtarget@
```

Note the initial cap!
Combination of built-in variables and your own:

```
set( libtarget auxlib )
```

Installing the pc file

```
install(
    FILES ${CMAKE_CURRENT_BINARY_DIR}/${PROJECT_NAME}.pc
    DESTINATION share/pkgconfig
)
```

Install while installing

```
include(ExternalProject)
include(ExternalProject)
ExternalProject_Add(googletest
  GIT_REPOSITORY    https://github.com/google/googletest.git
  GIT_TAG          master
  SOURCE_DIR        "${CMAKE_BINARY_DIR}/googletest-src"
  BINARY_DIR        "${CMAKE_BINARY_DIR}/googletest-build"
  CONFIGURE_COMMAND ""
  BUILD_COMMAND     ""
  INSTALL_COMMAND   ""
  TEST_COMMAND      ""
)
```