# The pkgconfig ecosystem

Victor Eijkhout

Fall 2022



# Justification

`Pkgconfig` is a *de facto* discovery mechanism for CMake packages. We discuss:

- how to discover package
- how to make your package discoverable

# Using packages through pkgconfig



# Problem

You want to install a package/application
. . . which needs 2 or 3 other packages.

```
cmake \
    -D PACKAGE1_INC=/users/my/package1/include \
    -D PACKAGE1_LIB=/users/my/package1/lib \
    -D PACKAGE2_INC=/users/my/package2/include/packaage \
    -D PACKAGE2_LIB=/users/my/package2/lib64 \
    ../newpackage
```

Can this be make simpler?

# Finding packages with ‘pkg config’

- Many packages come with a `package.pc` file
- Add that location to `PKG_CONFIG_PATH`
- That defines variables in your own cmake file

Example: PETSc
add `$PETSC_DIR/$PETSC_ARCH/lib/pkgconfig` to config path, then

```
find_package( PkgConfig REQUIRED )
pkg_check_modules( PETSC REQUIRED petsc )
target_include_directories(
        program PUBLIC
        ${PETSC_INCLUDE_DIRS} )
```

# Eigen

```
1  cmake_minimum_required( VERSION 3.12 )
2  project( eigentest )
3
4  find_package( PkgConfig REQUIRED )
5  pkg_check_modules( EIGEN REQUIRED eigen3 )
6
7  add_executable( eigentest eigentest.cxx )
8  target_include_directories(
9        eigentest PUBLIC
10       ${EIGEN_INCLUDE_DIRS})
```

# Where do these settings come from?

```
find $TACC_EIGEN_DIR -name \*.pc
${TACC_EIGEN_DIR}/share/pkgconfig/eigen3.pc
```

# Pkgconfig outside CMake

`.pc` files are also useful outside CMake:

```
$ pkg-config --cflags tbb
-I/opt/intel//oneapi/tbb/latest/lib/pkgconfig/../..//include
$ pkg-config --libs tbb
-L/opt/intel//oneapi/tbb/latest/lib/pkgconfig/../..//lib/intel64
    /gcc4.8 -ltbb
```

(useful for packages where there is no module)



# 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","/work2/00434/eijkhout/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@
```

Combination of built-in variables and your own:

```
set( libtarget auxlib )
```