An Introduction to CMake

Jeff Tranter <jtranter@ics.com>

February, 2019

Agenda

What is CMake?
Features
A Simple Example
A More Complex Example
Qt Support
Qt Example
Qt Creator Support
CTest and CPack
Misc. Tips
Summary
Q&A
References

What is CMake?

- Cross-platform build system
- Sits on top of and leverages native build system
- Funded by KitWare
- Written in C++
- Support for Qt
- Current version 3.13.3

Features

- Supports:
  - in place and out of place builds
  - enabling several builds from the same source tree
  - cross-compilation
- support for executables, static and dynamic libraries, generated files
- support for common build systems and SDKs

A Simple Example

CMakeLists.txt:

```
cmake_minimum_required(VERSION 3.0)
project(Demo1)
add_executable(Demo1 demo1.cpp)
```

A Simple Example

demo1.cpp:

```
#include <iostream>

int main()
{
  std::cout << "Hello, world!" << std::endl;
  return 0;
}
```

A Simple Example

In source build:

```
% cd demo1
% ls
CMakeLists.txt  demo1.cpp
```

A Simple Example

```
% cmake .
-- The C compiler identification is GNU 7.3.0
-- The CXX compiler identification is GNU 7.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tranter/git/jtranter/webinars/CMake/demo1
```

A Simple Example

```
% make
Scanning dependencies of target Demo1
[ 50%] Building CXX object
CMakeFiles/Demo1.dir/demo1.cpp.o
[100%] Linking CXX executable Demo1
[100%] Built target Demo1

% ./Demo1
Hello, world!
```

A Simple Example

Generated files:

CMakeCache.txt
CMakeFiles/
cmake_install.cmake
Demo1
Makefile

A Simple Example

Out of source build:

- `cd ..`
- `mkdir build`
- `cd build`

A Simple Example

```
% cmake ../demo1
-- The C compiler identification is GNU 7.3.0
-- The CXX compiler identification is GNU 7.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tranter/git/jtranter/webinars/CMake/build
```

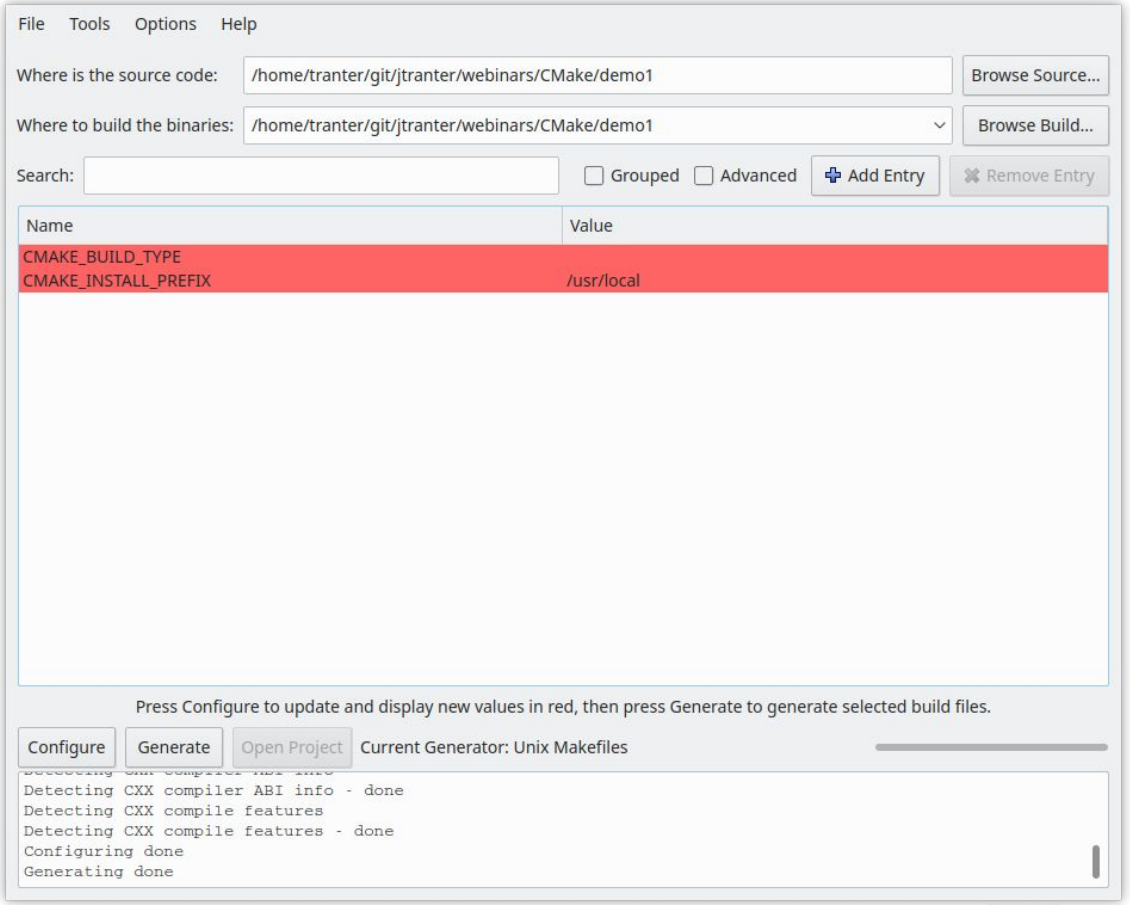
A Simple Example

```
% make
Scanning dependencies of target Demo1
[ 50%] Building CXX object
CMakeFiles/Demo1.dir/demo1.cpp.o
[100%] Linking CXX executable Demo1
[100%] Built target Demo1

% ./Demo1
Hello, world!
```

A Simple Example

Using cmake-gui:

```
% cd demo1
% cmake-gui .
```



A More Complex Example

To add more source files:

```
add_executable(Demo2 demo2.cpp hello.cpp)
```

Or if you have many files:

```
add_executable(Demo2
  demo2.cpp
  hello.cpp
  foo.cpp
  bar.cpp
  baz.cpp
)
```

A More Complex Example

Add a generated config file, in CMakeLists.txt:

```
# The version number.
set(Demo2_VERSION_MAJOR 1)
set(Demo2_VERSION_MINOR 0)

# Configure a header file to pass some of the CMake settings to source code
configure_file(
  "${PROJECT_SOURCE_DIR}/config.h.in"
  "${PROJECT_BINARY_DIR}/config.h"
  )

# Add the binary tree to the search path for include files
# so that we will find config.h
include_directories("${PROJECT_BINARY_DIR}")
```

A More Complex Example

config.h.in:

```
// The configured options and settings for Demo2
#define Demo2_VERSION_MAJOR @Demo2_VERSION_MAJOR@
#define Demo2_VERSION_MINOR @Demo2_VERSION_MINOR@
```

A More Complex Example

After running cmake, generated config.h:

```
// The configured options and settings for Demo2
#define Demo2_VERSION_MAJOR 1
#define Demo2_VERSION_MINOR 0
```

A More Complex Example

hello.h:

```
void hello();
```

hello.cpp:

```
#include <iostream>

void hello()
{
  std::cout << "Hello, world!" << std::endl;
}
```

A More Complex Example

demo2.cpp:

```
#include <iostream>
#include "hello.h"
#include "config.h"

int main()
{
  std::cout << "Demo2 version " << Demo2_VERSION_MAJOR
            << "." << Demo2_VERSION_MINOR << std::endl;
  hello();
  return 0;
}
```

A More Complex Example

Add a shared library:

- create mathlib folder for library functions
- create mathlib/mysqrt.cpp, mathlib/mysqrt.h

mathlib/CMakeLists.txt:

```
# Build a library of math functions
project(mathlib)
add_library(mathlib SHARED mysqrt.cpp)
```

A More Complex Example

Additions to top level CMakeLists.txt:

```
# Add library to include path
include_directories("${PROJECT_SOURCE_DIR}/mathlib")
add_subdirectory(mathlib)

# Link with math library
target_link_libraries(Demo2 mathlib)
```

A More Complex Example

Additions to demo2.cpp (highlighted):

```
#include <iostream>
#include "hello.h"
#include "config.h"
#include "mysqrt.h"

int main()
{
  std::cout << "Demo2 version " << Demo2_VERSION_MAJOR << "." << Demo2_VERSION_MINOR <<
std::endl;

  hello();

  for (double n = 1; n <= 10; n++) {
      std::cout << "mysqrt(" << n << ") = " << mysqrt(n) << std::endl;
  }

  return 0;
}
```

A More Complex Example

Add support for install. Additions to CMakeLists.txt:

```
# Add the install targets
install(TARGETS Demo2 DESTINATION bin)
install(FILES "${PROJECT_BINARY_DIR}/config.h" DESTINATION include)
```

Additions to mathlib/CMakeLists.txt:

```
# Install targets
install(TARGETS mathlib DESTINATION lib)
install(FILES mysqrt.h DESTINATION include)
```

A More Complex Example

Building it (out of source):

```
% mkdir build
% cd build/
% cmake ../demo2
-- The C compiler identification is GNU 7.3.0
-- The CXX compiler identification is GNU 7.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tranter/git/jtranter/webinars/CMake/build
```

A More Complex Example

```
% make
Scanning dependencies of target mathlib
[ 20%] Building CXX object mathlib/CMakeFiles/mathlib.dir/mysqrt.cpp.o
[ 40%] Linking CXX shared library libmathlib.so
[ 40%] Built target mathlib
Scanning dependencies of target Demo2
[ 60%] Building CXX object CMakeFiles/Demo2.dir/demo2.cpp.o
[ 80%] Building CXX object CMakeFiles/Demo2.dir/hello.cpp.o
[100%] Linking CXX executable Demo2
[100%] Built target Demo2
```

A More Complex Example

```
% ./Demo2
Demo2 version 1.0
Hello, world!
mysqrt(1) = 1
mysqrt(2) = 1.41421
mysqrt(3) = 1.73205
mysqrt(4) = 2
mysqrt(5) = 2.23607
mysqrt(6) = 2.44949
mysqrt(7) = 2.64575
mysqrt(8) = 2.82843
mysqrt(9) = 3
mysqrt(10) = 3.16228
```

A More Complex Example

```
% sudo make install
[ 40%] Built target mathlib
[100%] Built target Demo2
Install the project...
-- Install configuration: ""
-- Installing: /usr/local/bin/Demo2
-- Set runtime path of "/usr/local/bin/Demo2" to ""
-- Installing: /usr/local/include/config.h
-- Installing: /usr/local/lib/libmathlib.so
-- Installing: /usr/local/include/mysqrt.h
```

A More Complex Example

```
% ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake
config.h  Demo2  Makefile  mathlib

% ls mathlib
CMakeFiles  cmake_install.cmake  libmathlib.so  Makefile
```

Qt Support

- Part of standard CMake
- Requires CMake 3.1.0 or later
- Knows how to find Qt libraries
- Knows how to handle moc, UI files, resources

Qt Support

- CMake can find and use Qt 4 and Qt 5 libraries
- Automatically invokes moc, uic, rcc as needed
- AUTOMOC property controls automatic generation of moc
- AUTOUIC property controls automatic invocation of uic for UI files
- AUTORCC property controls automatic invocation of rcc for resource (.qrc) files

Qt Example

Simple Qt Creator wizard generated application:

main.cpp mainwindow.cpp mainwindow.h mainwindow.ui

Qt Example

CMakeLists.txt file:

```
cmake_minimum_required(VERSION 3.1.0)

project(Demo3)

# Find includes in corresponding build directories
set(CMAKE_INCLUDE_CURRENT_DIR ON)

# Instruct CMake to run moc automatically when needed
set(CMAKE_AUTOMOC ON)

# Create code from a list of Qt designer ui files
set(CMAKE_AUTOUIC ON)
```

Qt Example

CMakeLists.txt file (continued):

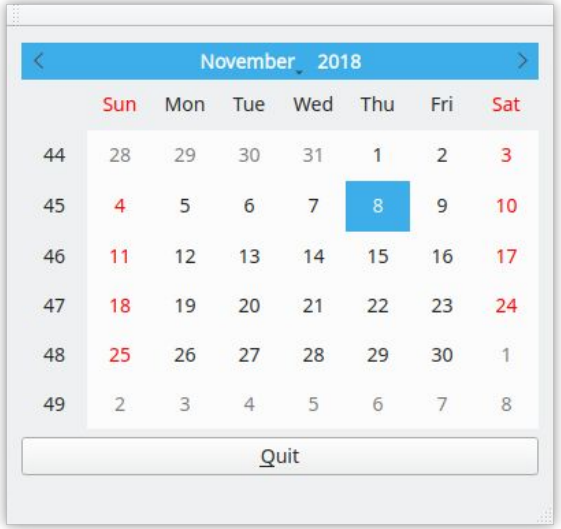
```
# Find the QtWidgets library
find_package(Qt5Widgets CONFIG REQUIRED)

# Populate a CMake variable with the sources
set(demo3_SRCS
    mainwindow.ui
    mainwindow.cpp
    main.cpp
)

# Tell CMake to create the Demo3 executable
add_executable(Demo3 WIN32 ${demo3_SRCS})

# Use the Widgets module from Qt 5
target_link_libraries(Demo3 Qt5::Widgets)
```

Qt Example

```
% cmake ../demo3
-- The C compiler identification is GNU 7.3.0
-- The CXX compiler identification is GNU 7.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tranter/git/jtranter/webinars/CMake/build
```

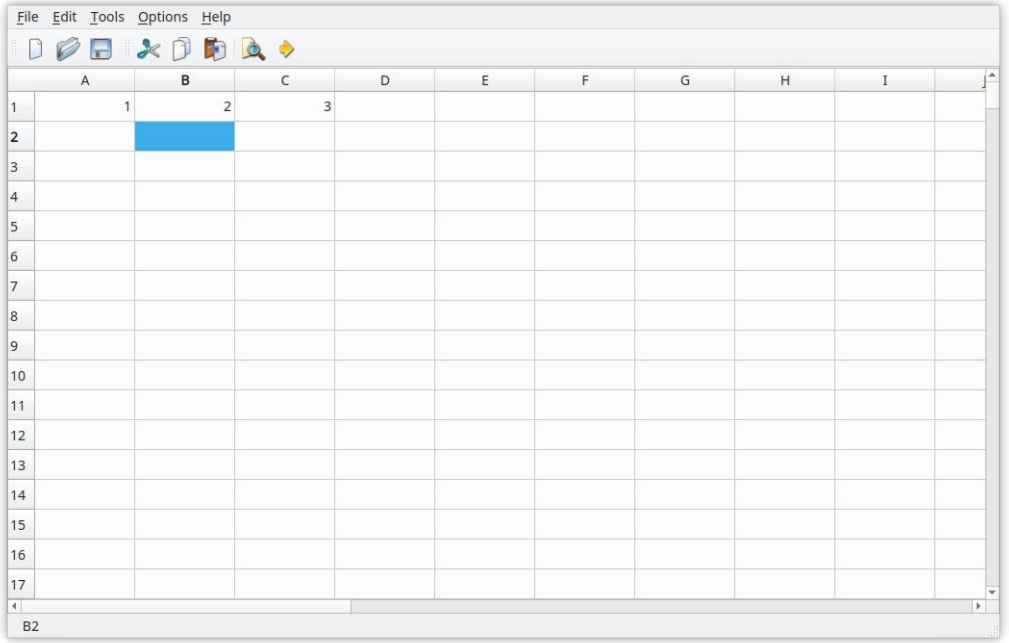
Qt Example

```
% make
Scanning dependencies of target Demo3_autogen
[ 20%] Automatic MOC and UIC for target Demo3
[ 20%] Built target Demo3_autogen
Scanning dependencies of target Demo3
[ 40%] Building CXX object CMakeFiles/Demo3.dir/mainwindow.cpp.o
[ 60%] Building CXX object CMakeFiles/Demo3.dir/main.cpp.o
[ 80%] Building CXX object CMakeFiles/Demo3.dir/Demo3_autogen/mocs_compilation.cpp.o
[100%] Linking CXX executable Demo3
[100%] Built target Demo3

% ./Demo3
```



Qt Example

- Larger spreadsheet program
- Qt 5 port of program from *C++ GUI Programming with Qt 4* book
- Widget-based, 13 source files, 2 UI files, resource file, 1300 LOC



Qt Example

Original qmake project file spreadsheet.pro:

```
lessThan(QT_MAJOR_VERSION, 5): error(This project requires Qt 5 or later)
TEMPLATE      = app
QT += widgets
HEADERS       = cell.h \
                finddialog.h \
                gotocelldialog.h \
                mainwindow.h \
                sortdialog.h \
                spreadsheet.h
SOURCES       = cell.cpp \
                finddialog.cpp \
                gotocelldialog.cpp \
                main.cpp \
                mainwindow.cpp \
                sortdialog.cpp \
                spreadsheet.cpp
FORMS         = gotocelldialog.ui \
                sortdialog.ui
RESOURCES     = spreadsheet.qrc
```

Qt Example

CMakeLists.txt file:

```
cmake_minimum_required(VERSION 3.1.0)

project(spreadsheet)

# Find includes in corresponding build directories
set(CMAKE_INCLUDE_CURRENT_DIR ON)

# Instruct CMake to run moc automatically when needed
set(CMAKE_AUTOMOC ON)

# Create code from a list of Qt designer ui files
set(CMAKE_AUTOUIC ON)

# Automatically handle resource files
set(CMAKE_AUTORCC ON)
```

Qt Example

CMakeLists.txt file (continued):

```
# Find the QtWidgets library
find_package(Qt5Widgets CONFIG REQUIRED)
# Populate a CMake variable with the sources
set(spreadsheet_SRCS
    cell.cpp
    finddialog.cpp
    gotocelldialog.cpp
    main.cpp
    mainwindow.cpp
    sortdialog.cpp
    spreadsheet.cpp
    spreadsheet.qrc
)
# Tell CMake to create the spreadsheet executable
add_executable(spreadsheet WIN32 ${spreadsheet_SRCS})

# Use the Widgets module from Qt 5
target_link_libraries(spreadsheet Qt5::Widgets)
```

Qt Example

```
% cmake ~/git/spreadsheet/
-- The C compiler identification is GNU 7.3.0
-- The CXX compiler identification is GNU 7.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tranter/git/jtranter/webinars/CMake/build/work
```

Qt Example

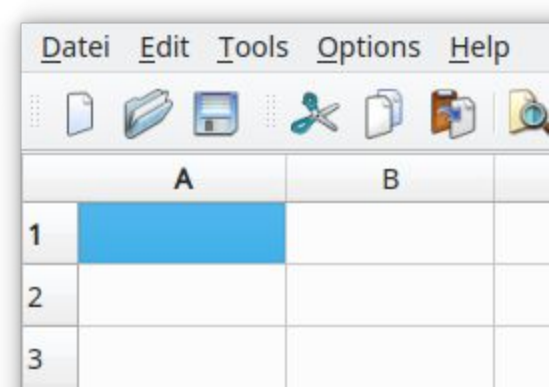
```
% make
Scanning dependencies of target spreadsheet_autogen
[  8%] Automatic MOC, UIC, and RCC for target spreadsheet
[  8%] Built target spreadsheet_autogen
[ 16%] Generating qrc_spreadsheet.cpp
Scanning dependencies of target spreadsheet
[ 25%] Building CXX object CMakeFiles/spreadsheet.dir/cell.cpp.o
[ 33%] Building CXX object CMakeFiles/spreadsheet.dir/finddialog.cpp.o
[ 41%] Building CXX object CMakeFiles/spreadsheet.dir/gotocelldialog.cpp.o
[ 50%] Building CXX object CMakeFiles/spreadsheet.dir/main.cpp.o
[ 58%] Building CXX object CMakeFiles/spreadsheet.dir/mainwindow.cpp.o
[ 66%] Building CXX object CMakeFiles/spreadsheet.dir/sortdialog.cpp.o
[ 75%] Building CXX object CMakeFiles/spreadsheet.dir/spreadsheet.cpp.o
[ 83%] Building CXX object CMakeFiles/spreadsheet.dir/qrc_spreadsheet.cpp.o
[ 91%] Building CXX object
CMakeFiles/spreadsheet.dir/spreadsheet_autogen/mocs_compilation.cpp.o
[100%] Linking CXX executable spreadsheet
[100%] Built target spreadsheet
```

Localization Support

```
FIND_PACKAGE(Qt5LinguistTools)

# Translation files
SET(TS_FILES
    spreadsheet_de.ts
    spreadsheet_fr.ts
 )

qt5_add_translation(QM_FILES ${TS_FILES})

# Tell CMake to create the spreadsheet executable
add_executable(spreadsheet WIN32 ${spreadsheet_SRCS} ${QM_FILES})
```

Localization Support

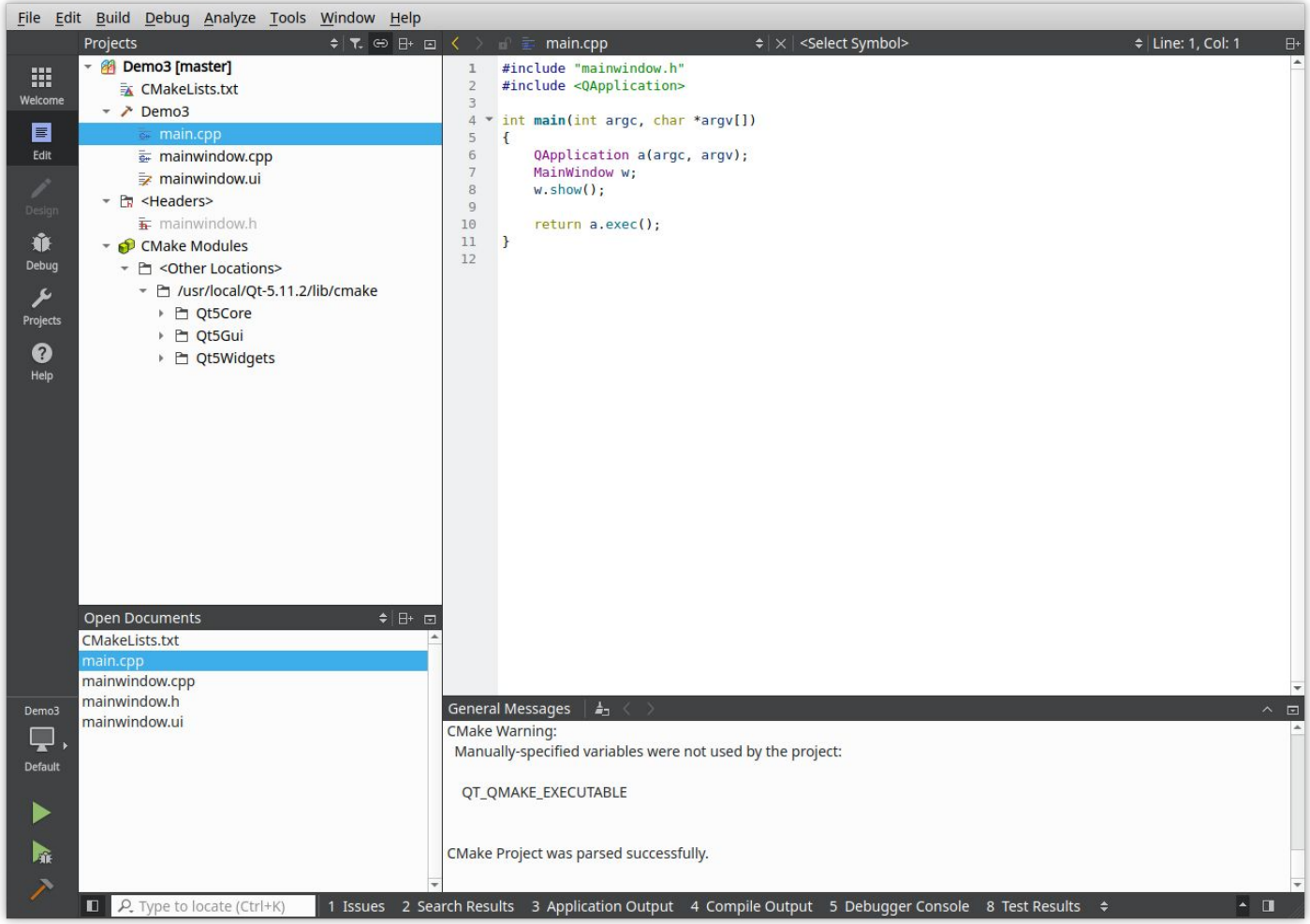
```
% cmake ../spreadsheet
...
[ 15%] Generating spreadsheet_fr.qm
Updating '/home/tranter/git/inactive/build/spreadsheet_fr.qm'...
    Generated 1 translation(s) (1 finished and 0 unfinished)
    Ignored 97 untranslated source text(s)
[ 23%] Generating spreadsheet_de.qm
Updating '/home/tranter/git/inactive/build/spreadsheet_de.qm'...
    Generated 1 translation(s) (1 finished and 0 unfinished)
    Ignored 97 untranslated source text(s)
...

% LANGUAGE=de ./spreadsheet
```



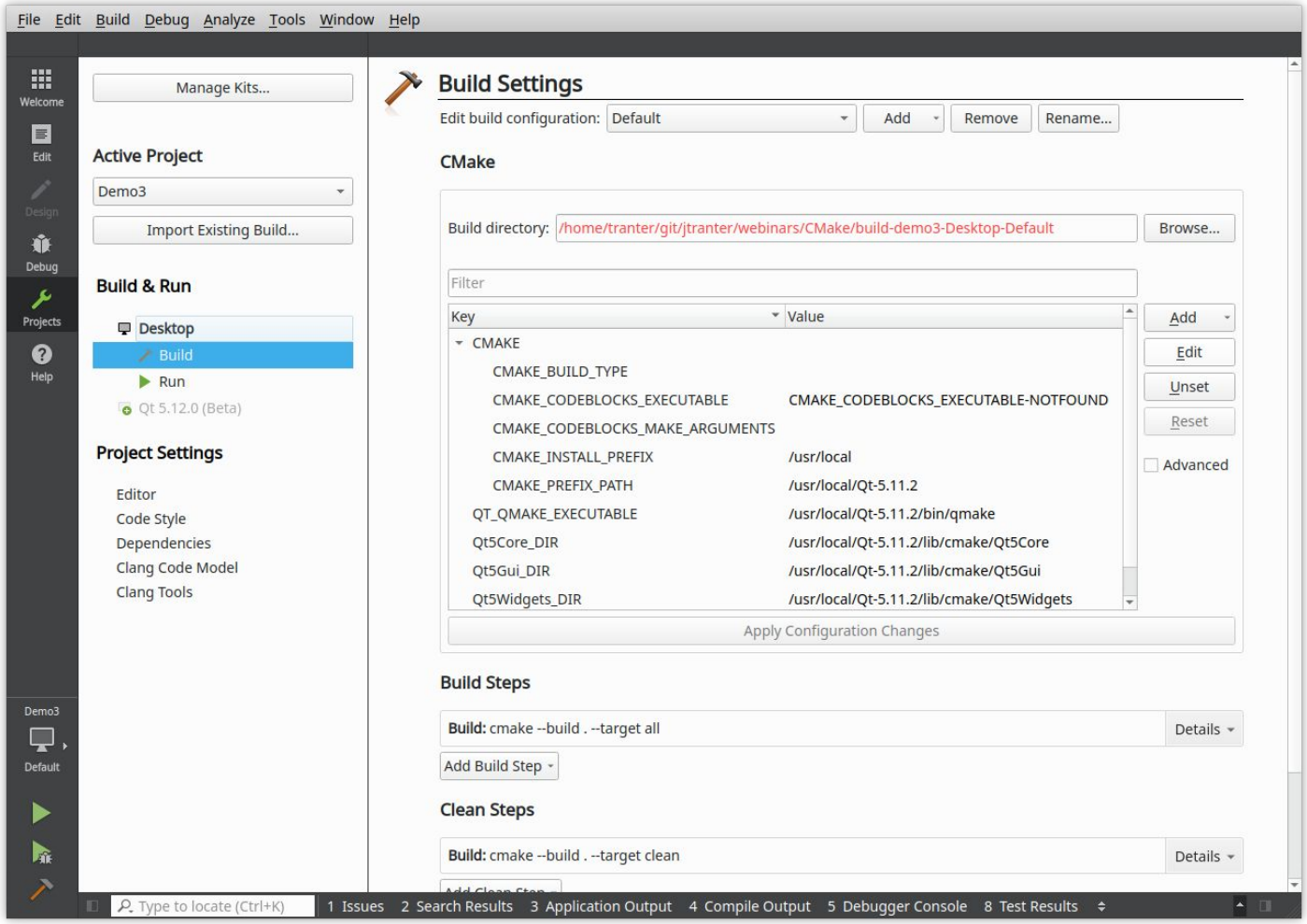
Qt Creator Support

- New project wizard can create a non-Qt C++ CMake project, but Qt-based projects all default to qmake
- You can open an existing CMake-based project and use it
- Can edit CMake variables in project settings pane
- Keeps it's settings in a CMakeLists.txt.user file (similar to .pro.user files with qmake)
- Make sure you enable the CMakeProjectManager plugin in Qt Creator
- Also check settings under Tools / Options... Kits/ CMake

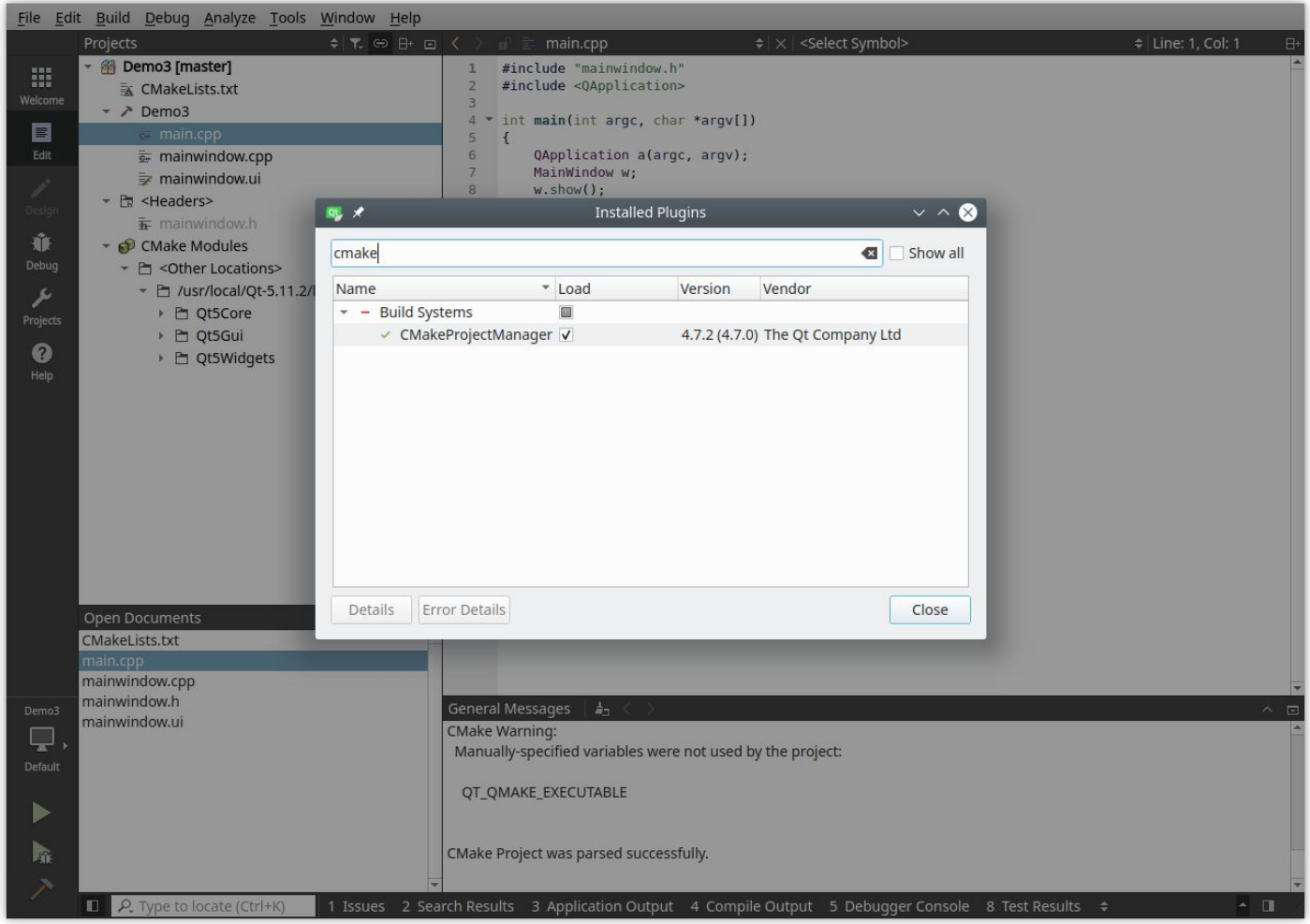
Qt Creator Support



Qt Creator Support



Qt Creator Support



CTest

- CMake has support for defining and running unit tests
- Uses the ctest program
- See the documentation for details

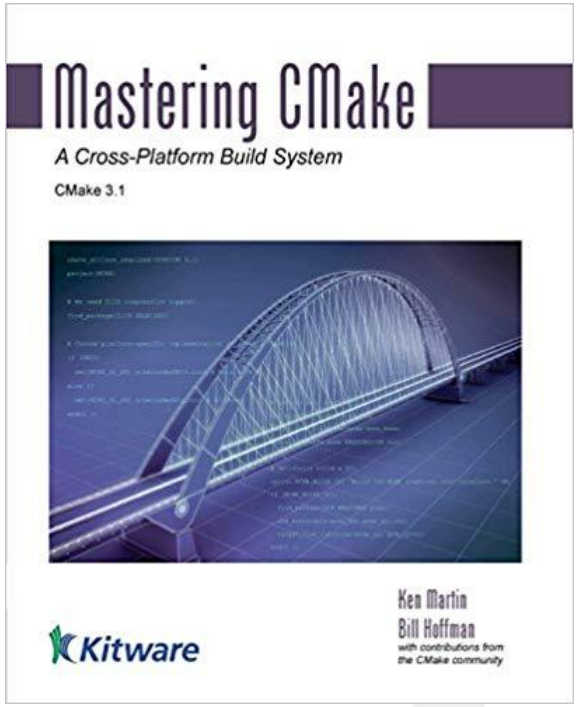
CPack

- Support for packaging applications via the "cpack" program and associated configuration files
- See the documentation for details

Misc. Tips

- For more verbose output when building: `make VERBOSE=1`
- Example of a sub-directories only project:

```
cmake_minimum_required(VERSION 3.0)
add_subdirectory(demo1)
add_subdirectory(demo2)
add_subdirectory(demo3)
```



Summary

- Use of CMake is increasing
- May be Qt's build system in Qt 6
- Relatively easy to use with Qt projects and Qt Creator

References

- https://en.wikipedia.org/wiki/CMake
- https://cmake.org/
- https://cmake.org/cmake-tutorial/
- https://cmake.org/cmake/help/latest/index.html
- https://cmake.org/download/
- https://cmake.org/cmake/help/latest/manual/cmake-qt.7.html#manual:cmake-qt(7)
- http://doc.qt.io/qt-5/cmake-manual.html
- http://doc.qt.io/qtcreator/creator-project-cmake.html
- https://github.com/tranter/webinars/tree/master/CMake
- https://wiki.qt.io/CMake_Port

Q&A

- Questions?
- Any tips to share?