CMake with Qt

Stephen Kelly
stephen.kelly@kdab.com

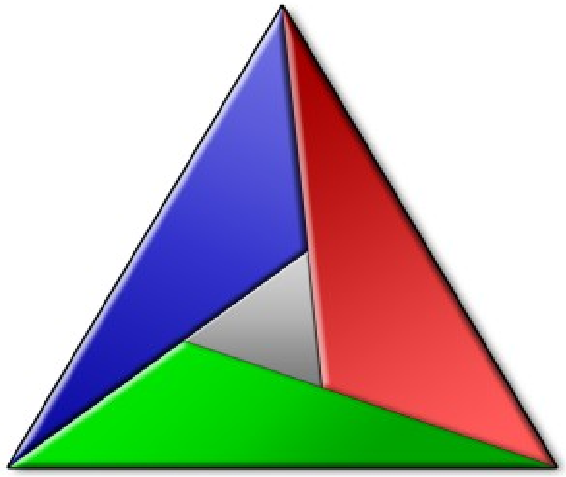
Qt Buildsystems

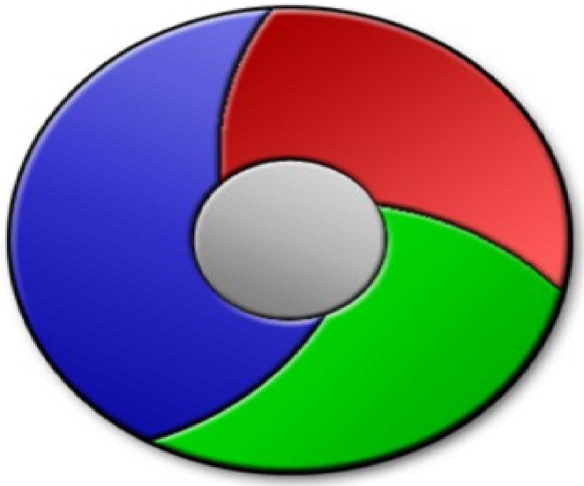
Qt 4	**Qt 5**
QMake	**CMake**

Why CMake?

- Finding dependencies
- Providing shared libraries
- Platform independence
  - Configure checks
  - Code generators
- Extendible
- Scalable

Related tools





CPack CTest CDash

Stages of building

Configure | Generate | moc | Compile | Link

Finding Dependencies

CMake today

```
find_package(Qt4 REQUIRED QtCore QtGui)


include_directories(${QT_INCLUDES})
add_definitions(${QT_DEFINITIONS})


add_executable(myexe WIN32 main.cpp)
target_link_libraries(myexe
    ${QT_QTCORE_LIBRARIES}
    ${QT_QTGUI_LIBRARIES}
)
```

CMake today

```
find_package(Qt5Widgets)

include_directories(${Qt5Widgets_INCLUDE_DIRS})
add_definitions(${Qt5Widgets_DEFINITIONS})
set(CMAKE_POSITION_INDEPENDENT_CODE ON)

add_executable(myexe WIN32 main.cpp)
target_link_libraries(myexe
    ${Qt5Widgets_LIBRARIES}
)
```

QMake (Qt 5)

```
TARGET = myexe
SOURCES = main.cpp
QT += widgets
```

CMake today

```
find_package(Qt5Widgets)

include_directories(${Qt5Widgets_INCLUDE_DIRS})
add_definitions(${Qt5Widgets_DEFINITIONS})
set(CMAKE_POSITION_INDEPENDENT_CODE ON)

add_executable(myexe WIN32 main.cpp)
target_link_libraries(myexe
    ${Qt5Widgets_LIBRARIES}
)
```

CMake tomorrow (2013)

```
find_package(Qt5Widgets)






add_executable(myexe WIN32 main.cpp)
target_link_libraries(myexe
    Qt5::Widgets
)
```

CMake tomorrow (2013)

```
find_package(Qt4)







add_executable(myexe WIN32 main.cpp)
target_link_libraries(myexe
    Qt4::Gui
)
```

Finding dependencies

```
find_package(Qt5Widgets)
find_package(boost …)

add_executable(dependent …)
target_link_libraries(dependent
        Qt5::Widgets
        boost::spirit
        boost::mpl)
```

The QTestLib problem

Finding dependencies

find_package(LibXml2)

- Libraries
- Header files
- Defines
- Version variables

find_package(Squish)

- Executables
- Test macros

Finding dependencies

- Treasure maps
  - FindLibXml2.cmake
  - FindSquish.cmake
  - FindQt4.cmake
  - Not part of upstream

- Local Guide
  - Qt5WidgetsConfig.cmake
  - PulseAudioConfig.cmake
  - Part of upstream package

Generating Code

Generated code

CMake

```
qt4_wrap_cpp(moc_files mywidget.h)
qt4_wrap_ui(ui_files mywidget.ui)
add_library(mylib ${sources}
                  ${moc_files}
                  ${ui_files})
```

Generated code

CMake

```
qt4_wrap_cpp(moc_files mywidget.h)
qt4_wrap_ui(ui_files mywidget.ui)
add_library(mylib ${sources}
                  ${moc_files}
                  ${ui_files})
```

QMake

```
SOURCES += mywidget.cpp
HEADERS += mywidget.h mycontrol.h
```

Automoc

CMake (2.8.8)

- No need for qt[4|5]_wrap_cpp
- #include “moc_mywidget.cpp”
  - Or don't!
- Finds Q_OBJECT in .h and .cpp files

Automoc

CMake (2.8.8)

```
qt4_wrap_ui(ui_files mywidget.ui)
add_library(mylib ${sources}
                  ${ui_files})
set_target_property(mylib AUTOMOC ON)
```

Automoc

CMake (2.8.8)

```
set(CMAKE_AUTOMOC ON)

qt4_wrap_ui(ui_files mywidget.ui)
add_library(mylib ${sources}
                  ${ui_files})
```

Extended code generation

```
qt4_create_translation(qm_files
        ${mylib_SRCS} mylib_de.ts
)
```

Extended code generation

```
qt4_generate_dbus_interface(mywidget.h
         com.kdab.mywidget.xml
)



qt4_add_dbus_interfaces(dbus_files
         com.kdab.mywidget.xml)
```

Platform independence

Platform checks

```
check_include_file(“limits.h” HAVE_LIMITS)

check_source_compiles(
  “#include <qglobal.h>
  #ifndef __PIC__
  #error \"Compile your code with -fPIC or -fPIE.\"
  #endif
  “
  HAVE_PIC
)
```

Creating Dependencies

Providing dependencies

```
install(TARGETS mylib
        EXPORT myexports …)

install(TARGETS otherlib
        EXPORT myexports …)

install(EXPORT myexport …)
```

Providing dependencies

```
install(EXPORT myexport
        FILENAME mytargets.cmake)


…
# mylibConfig.cmake
include(mytargets.cmake)

mylib_helpful_macro(...)
```

Providing dependencies

```
# No Findmylib.cmake needed!
find_package(mylib)



add_executable(dependent …)
target_link_libraries(dependent
        mylib
        otherlib)
```

Imported targets

```
# Qt5GuiConfig.cmake

add_library(Qt5::Gui IMPORTED)
set_target_property(Qt5::Gui
 LOCATION
   “/opt/qt5/lib/Qt5Gui.so”
)
```

Imported targets

```
# Qt5GuiConfig.cmake

add_library(Qt5::Gui IMPORTED)
set_target_property(Qt5::Gui
 INTERFACE_INCLUDE_DIRECTORIES
  “/opt/qt5/include/Qt5Gui”
  “$<TARGET_INCLUDES:Qt5::Core>”
)
```

Imported targets

```
# Qt5GuiConfig.cmake

add_library(Qt5::Gui IMPORTED)
set_target_property(Qt5::Gui
 INTERFACE_INCLUDE_DIRECTORIES
  "/opt/qt5/include/Qt5Gui"
 "$<TARGET_PROPERTY:Qt5::Core,INTERFACE_INCLUDE_DIRECTRIES>"
)
```

Imported targets

```
# Qt5GuiConfig.cmake

add_library(Qt5::Gui IMPORTED)
set_target_property(Qt5::Gui
 INTERFACE_INCLUDE_DIRECTORIES
   “/opt/qt5/include/Qt5Gui”
   “$<TARGET_INCLUDES:Qt5::Core>”
)
```

qtmain.lib

QMake

```
CONFIG -= console
```

CMake (Qt 4)

```
add_executable(exec WIN32 …)
target_link_libraries(${QT_QTMAIN_LIBS})
```

qtmain.lib

CMake (Qt 5)

```
add_executable(exec WIN32 …)
target_link_libraries(${QT_QTMAIN_LIBS})
```

Imported targets

```
# Qt5GuiConfig.cmake


add_library(Qt5::Gui IMPORTED)
set_target_property(Qt5::Gui
 INTERFACE_LINK_LIBRARIES
  “$<TARGET_LIBRARIES:Qt5::Core>”
  “$<$<WIN32_EXECUTABLE>:Qt5::WinMain>”
)
```

Declarative expressions

```
add_executable(CoolApp …)
target_link_libraries(CoolApp
    Qt5::Widgets
    $<$<CONFIG:Debug>:Qt5::Test>
)
```

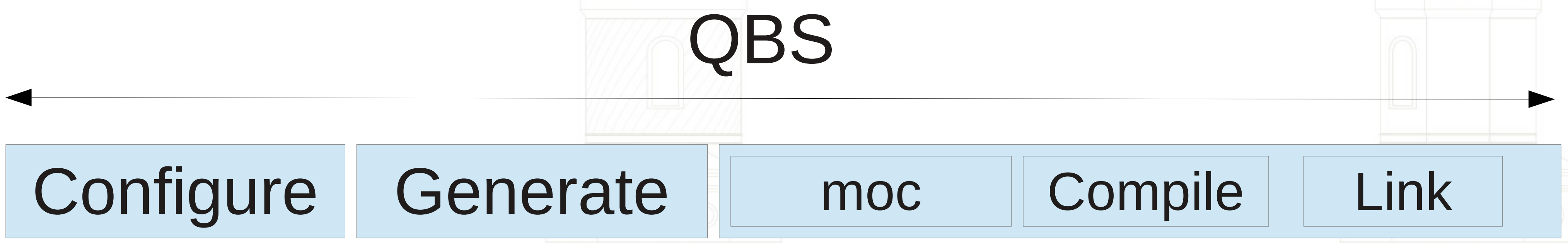
Declarative expressions

```
add_executable(CoolApp …)
set_target_property(CoolApp
    INCLUDE_DIRECTORIES
$<$<CONFIG:Debug>:/usr/include/valgrind>
)
```

- Object orientated
- Declarative

Qt Buildsystems

Qt 4	**Qt 5**
~~QMake~~ QBS	**CMake**


QBS
Configure
Generate
moc
Compile
Link

Questions?

Stephen Kelly
stephen.kelly@kdab.com