Qt + CMake + vcpkg

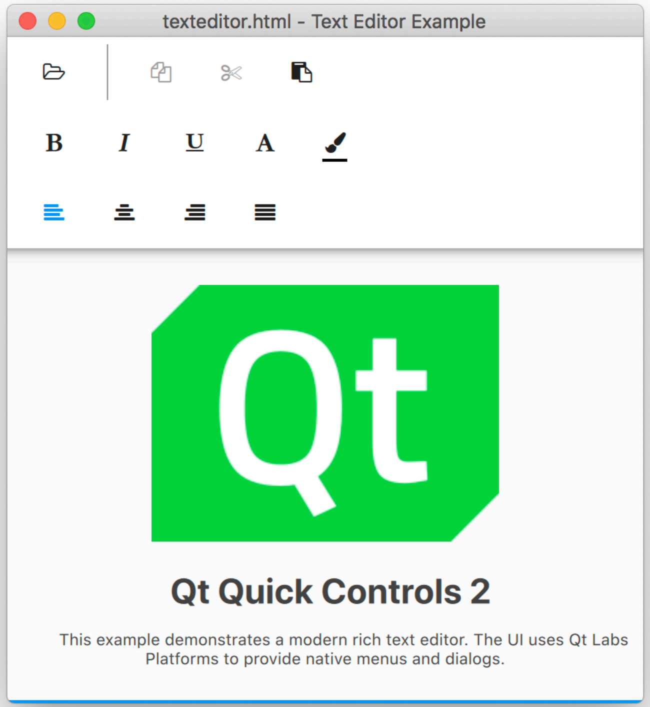
How to build a static Qt Quick Controls 2 app using CMake and vcpkg

05.11.2019

alcroito@Qt Company

Starting point

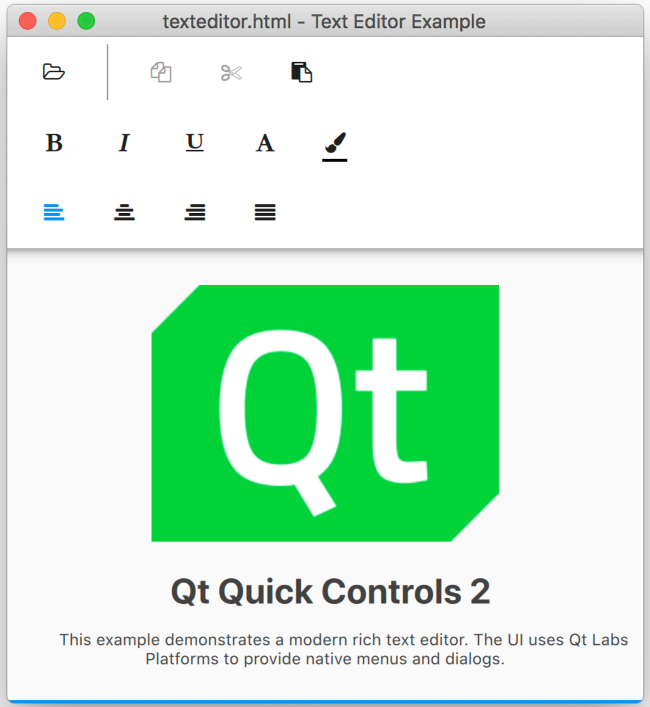
- Qt Quick Controls 2 example “texteditor”
- Mix of C++ and QML
- qmake project



This example demonstrates a modern rich text editor. The UI uses Qt Labs Platforms to provide native menus and dialogs.

Goals

- Build app using CMake
- Build it statically
- Use a third party library from vcpkg
- Target multiple platforms



This example demonstrates a modern rich text editor. The UI uses Qt Labs Platforms to provide native menus and dialogs.

Let’s dive in

.pro file

```
TEMPLATE = app
TARGET = texteditor
QT += quick quickcontrols2
qtHaveModule(widgets): QT += widgets

cross_compile: DEFINES += QT_EXTRA_FILE_SELECTOR="touch"

HEADERS += \
    documenthandler.h

SOURCES += \
    texteditor.cpp \
    documenthandler.cpp

RESOURCES += \
    texteditor.qrc
```

Initial CMakeLists.txt

```cmake
cmake_minimum_required(VERSION 3.15)

project(texteditor LANGUAGES CXX)

find_package(Qt5 5.14 REQUIRED COMPONENTS QuickControls2)
find_package(Qt5 5.14 COMPONENTS Widgets)

set(CMAKE_AUTOMOC ON)
set(CMAKE_AUTORCC ON)

add_executable(texteditor texteditor.cpp documenthandler.cpp texteditor.qrc)

if(CMAKE_CROSSCOMPILING)
    target_compile_definitions(texteditor PRIVATE QT_EXTRA_FILE_SELECTOR=\"touch\")
endif()

target_link_libraries(texteditor PRIVATE Qt5::Quick Qt5::QuickControls2)
if(TARGET Qt5::Widgets)
    target_link_libraries(texteditor PRIVATE Qt5::Widgets)
endif()
```

Minimum CMake version to build project

Find Qt libraries

Enable auto moc and rcc

Create executable

Mobile UI support

Use Qt libraries

Static builds?

Before Qt 5.14

```cmake
set(build_static TRUE)
if(build_static)
    # QuickTemplates2 package is also explicitly required for some reason.
    find_package(Qt5QuickTemplates2 REQUIRED)
    target_link_libraries(texteditor PRIVATE Qt5::QuickTemplates2)

    # Find all system libs
    find_library(FWSecurity Security)
    find_library(FWSystemConfiguration SystemConfiguration)
    find_library(FWCoreText CoreText)
    find_library(FWNetwork Network)
    find_library(FWCoreFoundation CoreFoundation)
    find_library(FWFoundation Foundation)
    find_library(FWCoreGraphics CoreGraphics)
    find_library(FWCoreServices CoreServices)
    find_library(FWAppKit AppKit)
    find_library(FWCoreVideo CoreVideo)
    find_library(FWIOKit IOKit)
    find_library(FWIOSurface IOSurface)
    find_library(FWCarbon Carbon)
    find_library(FWMetal Metal)
    find_library(FWQuartzCore QuartzCore)

    find_library(CUPS_LIBRARIES NAMES cups )
    find_package(OpenGL)
    find_package(ZLIB)

    # Link against qt provided 3rd party libs
    target_link_libraries(texteditor PRIVATE ${qt_dir}/lib/libqtharfbuzz.a)
    target_link_libraries(texteditor PRIVATE ${qt_dir}/lib/libqtlibpng.a)
```

```cmake
    target_link_libraries(texteditor PRIVATE ${qt_dir}/lib/libqtpcre2.a)
    target_link_libraries(texteditor PRIVATE ${qt_dir}/lib/libqtfreetype.a)

    # Link against the various platform support plugins
    target_link_libraries(texteditor PRIVATE ${qt_dir}/lib/libQt5FontDatabaseSupport.a)
    target_link_libraries(texteditor PRIVATE ${qt_dir}/lib/libQt5PrintSupport.a)
    target_link_libraries(texteditor PRIVATE ${qt_dir}/lib/libQt5GraphicsSupport.a)
    target_link_libraries(texteditor PRIVATE ${qt_dir}/lib/libQt5ClipboardSupport.a)
    target_link_libraries(texteditor PRIVATE ${qt_dir}/lib/libQt5ThemeSupport.a)
    target_link_libraries(texteditor PRIVATE ${qt_dir}/lib/libQt5AccessibilitySupport.a)

    # Link against the system libs
    target_link_libraries(texteditor PRIVATE ${FWSecurity} ${FWSystemConfiguration} ${FWCoreText} ${FWNetwork} ${FWCoreFoundation} ${FWCoreGraphics})
    target_link_libraries(texteditor PRIVATE ${FWIOKit} ${FWIOSurface} ${FWCarbon} ${FWMetal} ${FWQuartzCore})
    target_link_libraries(texteditor PRIVATE ${FWFoundation} ${FWCoreServices} ${FWAppKit} ${FWCoreVideo} OpenGL::GL ZLIB::ZLIB ${CUPS_LIBRARIES})

    # Link against the QPA plugin
    target_link_libraries(texteditor PRIVATE Qt5::QCocoaIntegrationPlugin)

    # Link against the Qt Quick plugins
    set(qml_plugin_dir "${qt_dir}/qml")
    target_link_libraries(texteditor PRIVATE ${qml_plugin_dir}/QtQuick.2/libqtquick2plugin.a)
    target_link_libraries(texteditor PRIVATE ${qml_plugin_dir}/QtQuick/Window.2/libwindowplugin.a)
    target_link_libraries(texteditor PRIVATE ${qml_plugin_dir}/QtQuick/Controls.2/libqtquickcontrols2plugin.a)
    target_link_libraries(texteditor PRIVATE ${qml_plugin_dir}/Qt/labs/platform/libqtlabsplatformplugin.a)
    target_link_libraries(texteditor PRIVATE ${qml_plugin_dir}/QtQuick/Templates.2/libqtquicktemplates2plugin.a)

    # Initialize the plugins
    target_sources(texteditor PRIVATE init_plugins.cpp)
endif()
```

After Qt 5.14+

```cmake
cmake_minimum_required(VERSION 3.15)

project(texteditor LANGUAGES CXX)

find_package(Qt5 5.14 REQUIRED COMPONENTS QuickControls2 QmlImportScanner)
find_package(Qt5 5.14 COMPONENTS Widgets)

set(CMAKE_AUTOMOC ON)
set(CMAKE_AUTORCC ON)

add_executable(texteditor texteditor.cpp documenthandler.cpp texteditor.qrc)

if(CMAKE_CROSSCOMPILING)
    target_compile_definitions(texteditor PRIVATE QT_EXTRA_FILE_SELECTOR=\"touch\")
endif()

target_link_libraries(texteditor PRIVATE Qt5::Quick Qt5::QuickControls2)
if(TARGET Qt5::Widgets)
    target_link_libraries(texteditor PRIVATE Qt5::Widgets)
endif()

# Import qml plugins for static builds.
qt5_import_qml_plugins(texteditor)
```

Regular plugins found automatically

Find qml import scanner

Import / build qml plugins

Third party libraries?

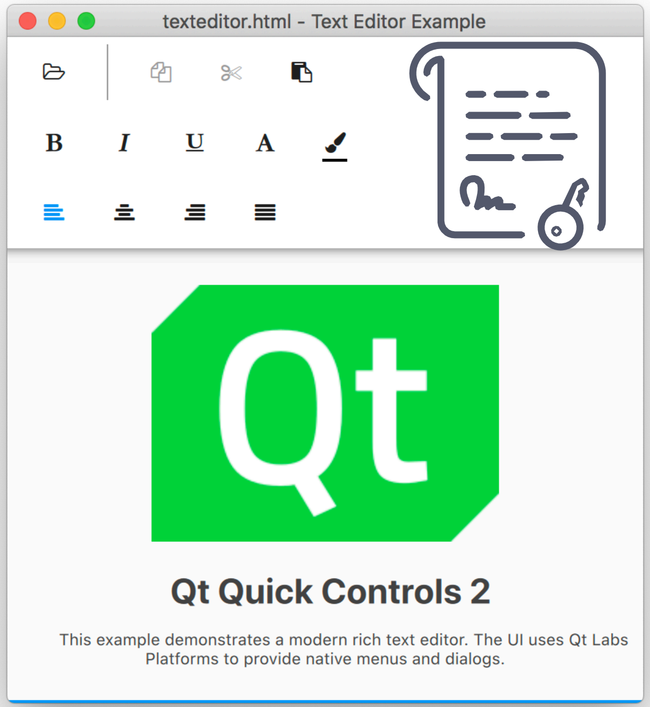
Third party libraries

› What's a useful feature for a text editor?

› Encrypted documents!

› Which C++ library provides encryption?

› Botan!



This example demonstrates a modern rich text editor. The UI uses Qt Labs Platforms to provide native menus and dialogs.

Botan

- BSD-licensed cryptographic library
- Written in C++
- Supports desktop and mobile platforms
- Where to get it?

vcpkg

- Contains recipes for building many libraries from source
- Targets multiple platforms (Windows / macOS / Linux) (in vcpkg lingo – triplets)
- Many packages integrate well with CMake
- Recipes written in CMake
- Incidentally, provides a recipe for Botan

Building Botan

```
cd <vcpkg_dir>
export VCPKG_DEFAULT_TRIPLET=x64-osx
./vcpkg install botan
```

Find Botan?

```cmake
# FindBotan.cmake

find_path(BOTAN_INCLUDE_DIRS NAMES botan/botan.h
          DOC "The botan include directory")

find_library(BOTAN_LIBRARIES NAMES botan botan-2
             DOC "The botan library")

include(FindPackageHandleStandardArgs)
find_package_handle_standard_args(Botan REQUIRED_VARS BOTAN_LIBRARIES BOTAN_INCLUDE_DIRS)

if(Botan_FOUND)
    add_library(Botan::Botan UNKNOWN IMPORTED)

    set_target_properties(Botan::Botan PROPERTIES
                          IMPORTED_LOCATION "${BOTAN_LIBRARIES}"
                          INTERFACE_INCLUDE_DIRECTORIES "${BOTAN_INCLUDE_DIRS}")
endif()

mark_as_advanced(BOTAN_LIBRARIES BOTAN_INCLUDE_DIRS)
```

Find headers

Find static libraries

Create CMake target

Assign location of headers and libraries to target

Use Botan!

```
cmake_minimum_required(VERSION 3.15)
include(cmake/app_utils.cmake)
setup_vcpkg_before_project()
project(encrypted_texteditor LANGUAGES CXX)

find_package(Qt5 5.14 REQUIRED COMPONENTS QuickControls2 QmlImportScanner)
find_package(Qt5 5.14 COMPONENTS Widgets)

list(APPEND CMAKE_MODULE_PATH "${CMAKE_CURRENT_SOURCE_DIR}/cmake")
find_package(Botan REQUIRED)

set(CMAKE_AUTOMOC ON)
set(CMAKE_AUTORCC ON)

add_executable(texteditor texteditor.cpp documenthandler.cpp texteditor.qrc)

if(CMAKE_CROSSCOMPILING)
    target_compile_definitions(texteditor PRIVATE QT_EXTRA_FILE_SELECTOR=\"touch\")
endif()

target_link_libraries(texteditor PRIVATE Qt5::Quick Qt5::QuickControls2 Botan::Botan)
if(TARGET Qt5::Widgets)
    target_link_libraries(texteditor PRIVATE Qt5::Widgets)
endif()

# Import qml plugins for static builds.
qt5_import_qml_plugins(texteditor)
```

Set up vcpkg

Find Botan

Use Botan

Set up vcpkg

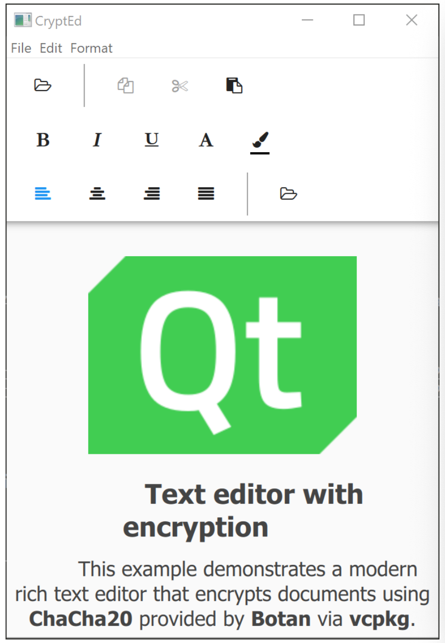
```cmake
function(setup_vcpkg_before_project)
    if(DEFINED ENV{VCPKG_ROOT})
        set(vcpkg_toolchain_path "$ENV{VCPKG_ROOT}/scripts/buildsystems/vcpkg.cmake")
        set(CMAKE_TOOLCHAIN_FILE "${vcpkg_toolchain_path}" CACHE STRING "" FORCE)
    endif()

    if(DEFINED ENV{VCPKG_DEFAULT_TRIPLET} AND NOT VCPKG_TARGET_TRIPLET)
        set(VCPKG_TARGET_TRIPLET "$ENV{VCPKG_DEFAULT_TRIPLET}" CACHE STRING "")
    endif()
endfunction()
```

Set vcpkg toolchain

Set vcpkg triplet

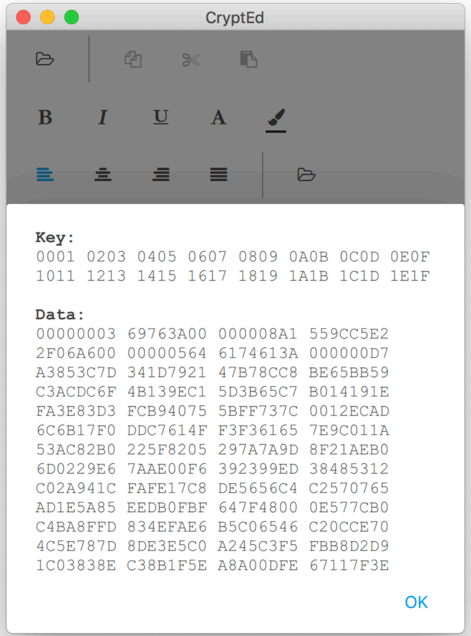
Multiple platforms?

Platforms

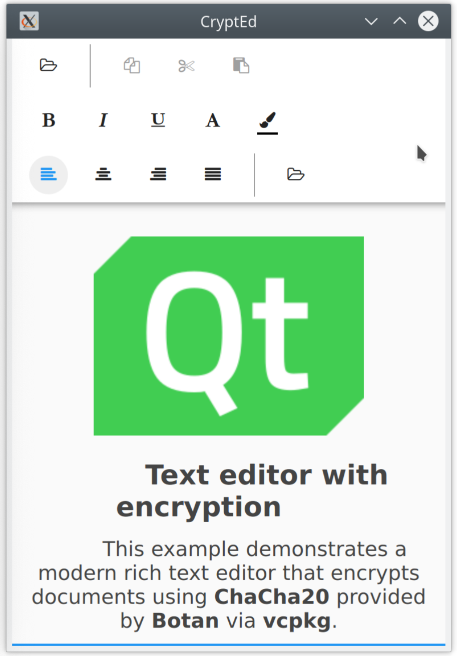


Text editor with encryption

This example demonstrates a modern rich text editor that encrypts documents using **ChaCha20** provided by **Botan** via **vcpkg**.

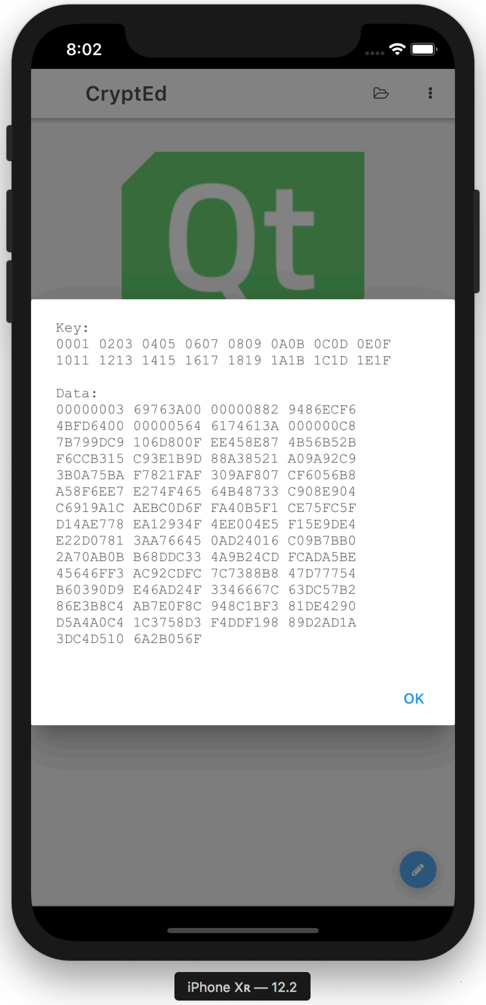


```
Key:
0001 0203 0405 0607 0809 0A0B 0C0D 0E0F
1011 1213 1415 1617 1819 1A1B 1C1D 1E1F

Data:
00000003 69763A00 000008A1 559CC5E2
2F06A600 00000564 6174613A 000000D7
A3853C7D 341D7921 47B78CC8 BE65BB59
C3ACDC6F 4B139EC1 5D3B65C7 B014191E
FA3E83D3 FCB94075 5BFF737C 0012ECAD
6C6B17F0 DDC7614F F3F36165 7E9C011A
53AC82B0 225F8205 297A7A9D 8F21AEB0
6D0229E6 7AAE00F6 392399ED 38485312
C02A941C FAFE17C8 DE5656C4 C2570765
AD1E5A85 EEDB0FBF 647F4800 0E577CB0
C4BA8FFD 834EFAE6 B5C06546 C20CCE70
4C5E787D 8DE3E5C0 A245C3F5 FBB8D2D9
1C03838E C38B1F5E A8A00DFE 67117F3E
```

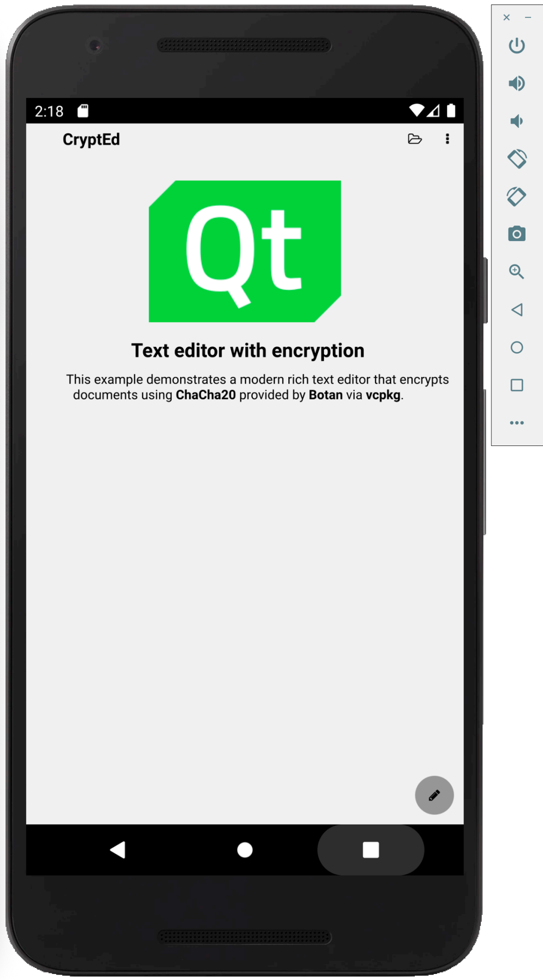


Text editor with encryption

This example demonstrates a modern rich text editor that encrypts documents using **ChaCha20** provided by **Botan** via **vcpkg**.



```
Key:
0001 0203 0405 0607 0809 0A0B 0C0D 0E0F
1011 1213 1415 1617 1819 1A1B 1C1D 1E1F

Data:
00000003 69763A00 00000882 9486ECF6
4BFD6400 00000564 6174613A 000000C8
7B799DC9 106D800F EE458E87 4B56B52B
F6CCB315 C93E1B9D 88A38521 A09A92C9
3B0A75BA F7821FAF 309AF807 CF6056B8
A58F6EE7 E274F465 64B48733 C908E904
C6919A1C AEBC0D6F FA40B5F1 CE75FC5F
D14AE778 EA12934F 4EE004E5 F15E9DE4
E22D0781 3AA76645 0AD24016 C09B7BB0
2A70AB0B B68DDC33 4A9B24CD FCADA5BE
45646FF3 AC92CDFC 7C7388B8 47D77754
B60390D9 E46AD24F 3346667C 63DC57B2
86E3B8C4 AB7E0F8C 948C1BF3 81DE4290
D5A4A0C4 1C3758D3 F4DDF198 89D2AD1A
3DC4D510 6A2B056F
```



Text editor with encryption

This example demonstrates a modern rich text editor that encrypts documents using **ChaCha20** provided by **Botan** via **vcpkg**.

Wait. Mobile too?

Mobile support

- No upstream support for mobile platforms in vcpkg
  - My fork contains modifications to build Botan for mobile (due to Botan not using CMake as its build system)
- Setting up CMake projects for iOS and Android is a trial and error process
  - No single source of truth for documentation
- Still possible!

iOS specifics

Certain things need to be done before the first *project()* call

```
set(MACOSX_BUNDLE_BUNDLE_NAME "CryptEd" PARENT_SCOPE)
set(MACOSX_BUNDLE_GUI_IDENTIFIER "io.qt.alcroito.crypted" PARENT_SCOPE)
set(MACOSX_BUNDLE_INFO_STRING "CryptEd" PARENT_SCOPE)
set(MACOSX_BUNDLE_LONG_VERSION_STRING "0.1.0" PARENT_SCOPE)
set(MACOSX_BUNDLE_SHORT_VERSION_STRING "0.1" PARENT_SCOPE)
set(MACOSX_BUNDLE_BUNDLE_VERSION "0.1" PARENT_SCOPE)
set_target_properties(${target} PROPERTIES MACOSX_BUNDLE_INFO_PLIST
                      "${CMAKE_CURRENT_SOURCE_DIR}/Info_ios.plist.in")

if(IOS)
    target_link_options(texteditor PRIVATE "-Wl,-e,_qt_main_wrapper")
endif()

set(CMAKE_OSX_DEPLOYMENT_TARGET "12.2" CACHE STRING "")
set(CMAKE_XCODE_ATTRIBUTE_CODE_SIGN_IDENTITY "iPhone Developer")

# Add the team identifier here, otherwise code-signing for the device won't work.
# Can also be set via CMake cache variable.
set(CMAKE_XCODE_ATTRIBUTE_DEVELOPMENT_TEAM "")

set(CMAKE_OSX_SYSROOT "iphoneos")
set(CMAKE_OSX_ARCHITECTURES "arm64" CACHE STRING "")
```

Specify usual Info.plist keys

Need to manually change application entry point

Set up code signing identity, development team, deployment target

Specify architecture and device or simulator

Android specifics

No static builds : (

```cmake
if(ANDROID)
    add_library("${target}" MODULE ${ARGN})
    # Not having this flag set might cause the executable to have main()
    # hidden and it can thus no longer be loaded through dlopen().
    set_property(TARGET "${target}" PROPERTY C_VISIBILITY_PRESET default)
    set_property(TARGET "${target}" PROPERTY CXX_VISIBILITY_PRESET default)
else()
    add_executable("${target}" WIN32 MACOSX_BUNDLE ${ARGN})
endif()

if(DEFINED CMAKE_TOOLCHAIN_FILE AND vcpkg_toolchain_path)
    get_filename_component(supplied_toolchain_file "${CMAKE_TOOLCHAIN_FILE}" ABSOLUTE)
    if(NOT supplied_toolchain_file STREQUAL vcpkg_toolchain_path)
        set(VCPKG_CHAINLOAD_TOOLCHAIN_FILE "${CMAKE_TOOLCHAIN_FILE}" CACHE STRING "")
    endif()
endif()
```

add_library(MODULE) instead of *add_executable*

Using vcpkg requires chainloading a toolchain

```bash
cd android_build
QT_PATH=/Users/alex/qt/qt514_android/
ANDROID_NDK_HOME=/Users/alex/android/sdk/ndk/20.0.5594570
ANDROID_SDK_HOME=/Users/alex/android/sdk
cmake .. -G"Unix Makefiles" -DANDROID_ABI=arm64-v8a \
      -DANDROID_SDK=$ANDROID_SDK_HOME
      -DANDROID_NDK=$ANDROID_NDK_HOME \
      -DANDROID_NATIVE_API_LEVEL=28 \
      -DCMAKE_TOOLCHAIN_FILE=/Users/alex/vcpkg/scripts/buildsystems/vcpkg.cmake \
      -DVCPKG_TARGET_TRIPLET=arm64-android \
      -DCMAKE_PREFIX_PATH=$QT_PATH -DCMAKE_FIND_ROOT_PATH=$QT_PATH \
      -DVCPKG_CHAINLOAD_TOOLCHAIN_FILE=$ANDROID_NDK_HOME/build/cmake/android.toolchain.cmake
make -j16
make -j16 VERBOSE=1 apk # Calls androiddeployqt
```

Need to specify many params when building

Minimum required versions

› Qt 5.14+

› CMake 3.15+ for iOS (lower for other platforms)

› Qt Creator 4.11+ for Android CMake integration (iOS CMake integration currently missing)

Links

- Full project source – https://github.com/alcroito/qt_world_summit_2019_cmake_vcpkg_app
  https://tiny.cc/qt_vcpkg_app
- Vcpkg fork for mobile support – https://github.com/alcroito/vcpkg/tree/qt_world_summit_2019_botan

- Slides included in Repo

Thanks. Questions?