Search

Building projects on the command line

This page explains how to configure and build existing projects. If you want to know how to create a Qt-based CMake project, see the documentation on how to get started with CMake.

To build a Qt project, CMake needs to know where the Qt installation is located. Usually this is done by setting the CMake variable CMAKE_PREFIX_PATH to Qt's installation prefix. If you are cross-compiling, see Cross-compiling for details on additional variables you will need to set.

If Qt is installed using the online installer, choose a Qt version within the top-level installation directory. For example, the following command shows how this is done on Windows:

```
cmake -DCMAKE_PREFIX_PATH=C:\Qt\6.4.0\msvc2019_64 -S <source-dir> -B <build-dir>
```

The and placeholders represent the source and build directories of your project.`<source-dir>` `<build-dir>`

CMake generators

CMake generates the necessary build system files that enable build tools such as GNU Make or Ninja to build your project.

CMake's default generator depends on the platform and build environment. For example on Windows, CMake generates Visual Studio project files if a Visual Studio environment is detected.

For a consistent developer experience on all platforms, use the or generator.`Ninja` `Ninja Multi-Config`

You can select the CMake generator either by setting the environment variable or using the argument:`CMAKE_GENERATOR` `-G`

```
cmake -G Ninja ...
```

qt-cmake

The script is a convenient alternative to configure your project. It eliminates the need for you to specify the . You can find it located in the directory of your Qt installation prefix. The script passes all parameters to CMake, so you can use it just like you would :`qt-cmake` `CMAKE_PREFIX_PATH` `bin\cmake`

```
C:\Qt\6.4.0\msvc2019_64\bin\qt-cmake -G Ninja -S <source-dir> -B <build-dir>
```

After the build system files are generated, your project is ready to be built:

```
cd <build-dir>
ninja
```

Qt DOCUMENTATION

```
cmake --build <build-dir>
```

Cross-compiling

Building your project for a platform that is different from your development machine is called cross-compiling. An example is building for Android (the target platform) on a Windows machine (the host platform).

Cross-compiling with CMake requires a toolchain file for most platforms. It also requires a Qt version for the development host, in addition to a Qt version for the target platform. For example, you need Qt for Windows and Qt for Android installed to cross-compile for Android on Windows.

从目标平台的 Qt 安装中使用，为该平台交叉编译项目：`qt-cmake`

```
<target-qt>/bin/qt-cmake -S <source-dir> -B <build-dir>
```

这将为目标平台配置项目。工具链文件会自动传递，并且可能还会设置其他特定于平台的变量。

指定自定义工具链文件

这脚本将 Qt 内部工具链文件传递给 CMake，此工具链文件设置了几个特定于Qt目标平台的变量。`qt-cmake`

在这种情况下，您可以指示通过设置变量来链接加载自定义工具链文件：`qt-cmakeQT_CHAINLOAD_TOOLCHAIN_FILE`

```
~/Qt/6.4.0/android_armv7/bin/qt-cmake -DQT_CHAINLOAD_TOOLCHAIN_FILE=<file-path> -S <source-dir> -B <build-dir>
```

这指示Qt的内部工具链文件也加载您的自定义工具链文件。

‹ 开始使用 导入的目标 ›

支持服务
专业服务
合作 伙伴
训练

支持中心
下载
秦特登录
联系我们
客户成功案例

社区

为Qt做贡献
论坛
维基
下载
市场

反馈 登录