Running qmake

The behavior of qmake can be customized when it is run by specifying various options on the command line. These allow the build process to be fine-tuned, provide useful diagnostic information, and can be used to specify the target platform for your project.

Command Syntax

The syntax used to run qmake takes the following simple form:

```
qmake [mode] [options] files
```

Note: If you installed Qt via a package manager, the binary may be `qmake6`.

Operating Modes

qmake supports two different modes of operation. In the default mode, qmake uses the information in a project file to generate a Makefile, but it is also possible to use qmake to generate project files. If you want to explicitly set the mode, you must specify it before all other options. The `mode` can be either of the following two values:

- `-makefile`
  qmake output will be a Makefile.
- `-project`
  qmake output will be a project file.

  Note: It is likely that the created file will need to be edited. For example, adding the QT variable to suit what modules are required for the project.

You can use the `options` to specify both general and mode-specific settings. Options that only apply to the Makefile mode are described in the Makefile Mode Options section, whereas options that influence the creation of project files are described in the Project Mode Options section.

Files

General Options

A wide range of options can be specified on the command line to qmake in order to customize the build process, and to override default settings for your platform. The following basic options provide help on using qmake, specify where qmake writes the output file, and control the level of debugging information that will be written to the console:

- `-help`
  qmake will go over these features and give some useful help.
- `-o file`
  qmake output will be directed to `file`. If this option is not specified, qmake will try to use a suitable file name for its output, depending on the mode it is running in.
  If '-' is specified, output is directed to stdout.
- `-d`
  qmake will output debugging information. Adding `-d` more than once increases verbosity.

The template used for the project is usually specified by the TEMPLATE variable in the project file. You can override or modify this by using the following options:

- `-t tmpl`
  qmake will override any set `TEMPLATE` variables with `tmpl`, but only *after* the .pro file has been processed.
- `-tp prefix`
  qmake will add `prefix` to the `TEMPLATE` variable.

The level of warning information can be fine-tuned to help you find problems in your project file:

- `-Wall`
  qmake will report all known warnings.
- `-Wnone`
  No warning information will be generated by qmake.
- `-Wparser`
  qmake will only generate parser warnings. This will alert you to common pitfalls and potential problems in the parsing of your project files.
- `-Wlogic`
  qmake will warn of common pitfalls and potential problems in your project file. For example, qmake will report multiple occurrences of files in lists and missing files.

Makefile Mode Options

```
qmake -makefile [options] files
```

In Makefile mode, qmake will generate a Makefile that is used to build the project. Additionally, the following options may be used in this mode to influence the way the project file is generated:

- `-after`
  qmake will process assignments given on the command line after the specified files.
- `-nocache`
  qmake will ignore the `.qmake.cache` file.

- `-cache file`
  qmake will use `file` as the cache file, ignoring any other .qmake.cache files found.
- `-spec spec`
  qmake will use `spec` as a path to platform and compiler information, and ignore the value of QMAKESPEC.

You may also pass qmake assignments on the command line. They are processed before all of the files specified. For example, the following command generates a Makefile from test.pro:

```
qmake -makefile -o Makefile "CONFIG+=test" test.pro
```

However, some of the specified options can be omitted as they are default values:

```
qmake "CONFIG+=test" test.pro
```

If you are certain you want your variables processed after the files specified, then you may pass the `-after` option. When this is specified, all assignments on the command line after the `-after` option will be postponed until after the specified files are parsed.

Project Mode Options

```
qmake -project [options] files
```

In project mode, qmake will generate a project file. Additionally, you may supply the following options in this mode:

- `-r`
  qmake will look through supplied directories recursively.
- -nopwd
  qmake will not look in your current working directory for source code. It will only use the specified `files`.

In this mode, the `files` argument can be a list of files or directories. If a directory is specified, it will be included in the DEPENDPATH variable, and relevant code from there will be included in the generated project file. If a file is given, it will be appended to the correct variable, depending on its extension. For example, UI files are added to FORMS, and C++ files are added to SOURCES.

You may also pass assignments on the command line in this mode. When doing so, these assignments will be placed last in the generated project file.

‹ Building Common Project Types Platform Notes ›

DOCUMENTATION

The Qt Company

Contact Us

Company

About Us
Investors
Newsroom
Careers
Office Locations

Licensing

Terms & Conditions
Open Source
FAQ

Support

Support Services
Professional Services
Partners
Training

For Customers

Support Center
Downloads
Qt Login
Contact Us
Customer Success

Community

Contribute to Qt
Forum
Wiki
Downloads
Marketplace

Feedback Sign In