搜索

变量

qmake 的基本行为受定义每个项目的生成过程的变量声明的影响。其中一些声明每个平台通用的资源，例如标头和源文件。其他用于自定义特定平台上编译器和链接器的行为。

特定于平台的变量遵循它们扩展或修改的变量的命名模式，但在其名称中包含相关平台的名称。例如，makespec 可用于指定每个项目需要链接的库列表，并用于扩展此列表。QMAKE_LIBSQMAKE_LIBS_X11

ANDROID_ABIS

注意：此变量仅适用于安卓目标。

指定安卓目标 ABI 的列表。有效值为：armeabi-v7a、arm64-v8a、x86、x86_64。

您可以将 ABI 作为 qmake 参数提供：

```
qmake ANDROID_ABIS="armeabi-v7a arm64-v8a"
```

注意：可以在 *.pro 文件中使用此变量，但是，不建议这样做，因为它将覆盖命令行上指定的任何 ABI。qmake

ANDROID_API_VERSION

注意：此变量仅适用于安卓目标。

指定安卓 API 级别编号。有关更多信息，请参阅Android 内部版本号。

ANDROID_APPLICATION_ARGUMENTS

注意：此变量仅适用于安卓目标

使用 the with 标应 "android.app.arguments" 为 Android 应用程序指定额外的命令行参数。这需要一串参数：AndroidManifest.xml

```
ANDROID_APPLICATION_ARGUMENTS = "arg1 arg2 arg3"
```

ANDROID_BUNDLED_JAR_DEPENDENCIES

> **注意**：此变量仅适用于安卓模块。

这在编写Qt模块时很有用。它以 aformat 指定模块使用的预捆绑依赖项列表，例如：.jar

```
ANDROID_BUNDLED_JAR_DEPENDENCIES += jar/Qt6Android.jar
```

ANDROID_DEPLOYMENT_DEPENDENCIES

> **注意**：此变量仅适用于安卓目标。

默认情况下，androiddeployqt会检测应用程序的依赖关系。但是，由于无法检测到插件的运行时使用情况，因此可能会出现误报，因为您的应用程序可能依赖于任何潜在依赖项的插件。如果要最小化 YOUR 的大小，可以使用 this 变量覆盖自动检测。这应该包含需要包含的所有Qt文件的列表，以及与Qt安装根目录相关的路径。APK

> **注意**：仅包含使用此变量指定的 Qt 文件。未能包含所有正确的文件可能会导致崩溃。确保文件以正确的加载顺序列出也很重要。此变量提供了一种完全覆盖自动检测的方法，因此，如果库列在其依赖项之前，则它将无法在某些设备上加载。

ANDROID_DEPLOYMENT_SETTINGS_FILE

> **注意**：此变量仅适用于安卓目标。

指定androiddeployqt所需的文件的路径。这将覆盖 qmake 生成的设置文件的路径，因此您必须确保提供有效的设置文件。android-deployment-settings.jsonandroidtestrunner

ANDROID_EXTRA_LIBS

> **注意**：此变量仅适用于安卓目标。

外部库的列表，这些库将复制到应用程序的文件夹中并在启动时加载。例如，这可用于在您的应用程序中启用 OpenSSL。有关更多信息，请参阅添加对 Android 的 OpenSSL 支持。libs

```
for (abi, ANDROID_ABIS): ANDROID_EXTRA_LIBS += $$PWD/$${abi}/library_name.so
```

否则，如果库名称中包含 ABI，请使用以下命令：

```
for (abi, ANDROID_ABIS): ANDROID_EXTRA_LIBS += $$PWD/library_name_$${abi}.so
```

ANDROID_EXTRA_PLUGINS

> **注意：**此变量仅适用于安卓目标。

指定C++的路径，您的应用程序必须捆绑这些插件或资源，但无法通过资产系统交付，例如 QML 插件。有了这个变量，androiddeployqt将确保所有内容都正确打包和部署。

ANDROID_EXTRA_PLUGINS必须指向构建额外插件的目录。此外，构建目录结构必须遵循类似于Qt插件的命名约定，即*plugins/<plugin name>*。

插件库的名称格式应为 libplugins_<type>_<*name*>_<*abi*>*.so*。为了实现这一点，插件pro文件可以定义如下：

```
TEMPLATE = lib
CONFIG += plugin

PLUGIN_TYPE = imageformats
DESTDIR = $$top_builddir/plugins/myplugin
TARGET = $$qt5LibraryTarget(myplugin, "plugins/$$PLUGIN_TYPE/")
```

top_builddir在 .qmake.conf 中定义为：

```
top_builddir=$$shadowed($$PWD)
```

这将确保将正确的名称重整应用于插件库（*plugins/myplugin/libplugins_imageformats_myplugin_armeabi-v7a.so*）。

然后，假设一个额外的图像格式插件*myplugin*构建为*$$DESTDIR/plugins/myplugin/*，以下内容确保它被正确打包：

```
ANDROID_EXTRA_PLUGINS += $$top_builddir/plugins
```

ANDROID_FEATURES

> **注意：**此变量仅适用于安卓模块

指定模块的功能列表：

```
ANDROID_FEATURES += android.hardware.location.gps
```

有关更多信息，请参阅Android <使用功能>文档。

ANDROID_LIB_DEPENDENCIES

> **注意：**此变量仅适用于安卓模块。

这在编写Qt模块时很有用。它指定模块使用的预构建依赖项的列表，例如：

```
ANDROID_LIB_DEPENDENCIES += \
    plugins/libplugins_platforms_qtforandroid.so
```

ANDROID_MIN_SDK_VERSION

> **注意：**此变量仅适用于安卓目标。

指定项目的最低 Android API 级别。默认情况下，此变量设置为 API 级别 23。

ANDROID_PACKAGE_SOURCE_DIR

> **注意：**此变量仅适用于安卓目标。

指定自定义 Android 程序包模板的路径。安卓软件包模板包含：

- 安卓清单.xml文件
- build.gradle 文件和其他 Gradle 脚本
- res/values/libs.xml 文件

此变量指定的路径可以在目录下包含自定义 Java 类。默认情况下，androiddeployqt工具将应用程序模板从 Qt for Android 安装路径复制到项目的构建目录中，然后将此变量指定的路径的内容复制到该目录上，覆盖任何现有文件。例如，您可以为应用程序进行自定义，然后将其直接放入此变量指定的目录中。`srcAndroidManifest.xml`

ANDROID_PERMISSIONS

> **注意：**此变量仅适用于安卓模块。

```
ANDROID_PERMISSIONS += android.permission.ACCESS_FINE_LOCATION
```

有关更多信息，请参阅Android <使用权限>文档。

ANDROID_TARGET_SDK_VERSION

> **注意**：此变量仅适用于安卓目标。

指定项目的目标 Android API 级别。默认情况下，此变量设置为 API 级别 30。

ANDROID_VERSION_CODE

> **注意**：此变量仅适用于安卓目标。

指定应用程序的版本号。有关更多信息，请参阅Android 应用版本控制。

ANDROID_VERSION_NAME

> **注意**：此变量仅适用于安卓目标。

将应用程序的版本指定为人类可读的字符串。有关更多信息，请参阅Android 应用版本控制。

配置

指定项目配置和编译器选项。这些值由 qmake 内部识别，具有特殊含义。

> **注意**：这些值区分大小写。

以下值控制编译器和链接器标志：CONFIG

选择	描述
释放	项目将在发布模式下生成。还指定了 ifis，最后一个生效。debug
调试	项目将在调试模式下生成。
debug_and_release	该项目已准备好*在调试和*发布模式下生成。
debug_and_release_target	默认情况下设置此选项。还设置了 ifis 后，调试和发布版本最终位于单独的调试和发布目录中。debug_and_release
build_all	指定 ifis，默认情况下，项目在调试和发布模式下生成。debug_and_release

	注意：不建议使用此选项。按照SUBDIRS变量文档中的说明指定依赖关系。
precompile_header	启用对在项目中使用预编译标头的支持。
precompile_header_c（仅限 MSVC）	启用对 C 文件使用预编译标头的支持。
warn_on	编译器应输出尽可能多的警告。还指定了 ifis，最后一个生效。warn_off
warn_off	编译器应输出尽可能少的警告。
异常	启用异常支持。默认设置。
exceptions_off	异常支持已禁用。
莱特克	链接时间码生成已启用。默认情况下，此选项处于关闭状态。
RTTI	已启用 RTTI 支持。默认情况下，使用编译器默认值。
rtti_off	RTTI 支持已禁用。默认情况下，使用编译器默认值。
STL	已启用 STL 支持。默认情况下，使用编译器默认值。
stl_off	STL 支持已禁用。默认情况下，使用编译器默认值。
线	线程支持已启用。当 CONFIG 包含时启用此功能，这是默认值。qt
no_utf8_source	指定项目的源文件不使用 UTF-8 编码。而是使用编译器默认值。
hide_symbols	将二进制文件中符号的默认可见性设置为隐藏。默认情况下，使用编译器默认值。
C99	已启用 C99 支持。如果编译器不支持 C99 或无法选择 C 标准，则此选项不起作用。默认情况下，使用编译器默认值。
C11	已启用 C11 支持。如果编译器不支持 C11 或无法选择 C 标准，则此选项不起作用。默认情况下，使用编译器默认值。
C17	C17（也称为 C18）支持已启用。如果编译器不支持 C17 或无法选择 C 标准，则此选项不起作用。默认情况下，使用编译器默认值。
C18	这是值的别名。c17
strict_c	禁用对 C 编译器扩展的支持。默认情况下，它们处于启用状态。
C++11	已启用 C++11 支持。如果编译器不支持 C++11 或无法选择C++标准，则此选项不起作用。默认情况下，支持处于启用状态。
C++14	已启用 C++14 支持。如果编译器不支持 C++14 或无法选择C++标准，则此选项不起作用。默认情况下，支持处于启用状态。
C++17	已启用 C++17 支持。如果编译器不支持 C++17，或者无法选择C++标准，则此选项不起作用。默认情况下，支持处于启用状态。
C++1Z	c++17 的过时别名。
C++20	已启用 C++20 支持。如果编译器不支持 C++20，或者无法选择C++标准，则此选项不起作用。默认情况下，支持处于禁用状态。
C++2A	c++20 的过时别名。
C++最新	启用了对编译器支持的最新C++语言标准的支持。默认情况下，此选项处于禁用状态。

depend_includepath	启用将 INCLUDEPATH 的值追加到 DEPENDPATH。默认设置。
lrelease	运行翻译和EXTRA_TRANSLATIONS中列出的所有文件。如果未设置 ifis，请将生成的 .qm 文件安装到QM_FILES_INSTALL_PATH中。使用 QMAKE_LRELEASE_FLAGS 将选项添加到 lrelease 调用。默认情况下未设置。 lreleaseembed_translations
embed_translations	在可执行文件的QM_FILES_RESOURCE_PREFIX下嵌入生成的翻译。也需要设置。默认情况下未设置。 lreleaselrelease
create_libtool	为当前构建的库创建一个 libtool .la 文件。
create_pc	为当前构建的库创建一个 pkg-config .pc 文件。
no_batch	仅 NMake：关闭生成 NMake 批处理规则或推理规则。
skip_target_version_ext	禁止在 Windows 上取消附加到 DLL 文件名的自动版本号。
suppress_vcproj_warnings	禁止显示 VS 项目生成器的警告。
WinDeployqt	链接后自动调用 windeployqt，并将输出添加为部署项。
dont_recurse	禁止当前子项目的 qmake 递归。
no_include_pwd	不要将当前目录添加到包含路径。
compile_included_sources	默认情况下，qmake 不会编译其他源文件中包含的源文件。此选项禁用此行为。

当您使用选项（这是Windows下的默认设置）时，项目将被处理三次：一次用于生成"元"Makefile，另外两次用于生成Makefile.Debug和Makefile.Release。 debug_and_release

在后一关时，附加了相应的选项。这使得执行特定于构建的任务成为可能。例如： build_passdebugreleaseCONFIG

```
build_pass:CONFIG(debug, debug|release) {
    unix: TARGET = $$join(TARGET,,,_debug)
    else: TARGET = $$join(TARGET,,,d)
}
```

作为手动编写构建类型条件的替代方法，某些变量除了常规QMAKE_LFLAGS之外，还提供特定于构建的变体，例如QMAKE_LFLAGS_RELEASE。这些应在可用时使用。

元 Makefile 使子构建可通过 theandtargets 调用，并通过 thetarget 组合构建。使用该选项时，组合生成是默认值。否则，集合 （，） 中的最后一个指定选项将确定默认值。在这种情况下，您可以显式调用 target 以同时构建两个配置： debugreleaseallbuild_allCONFIGCONFIGdebugreleaseall

```
make all
```

注意：在制作Visual Studio和Xcode项目时，细节略有不同。

链接库时，qmake 依靠底层平台来了解该库链接的其他库。但是，如果静态链接，除非我们使用以下选项，否则 qmake 将不会获得此信息： CONFIG

	将保存有关库的元信息（有关详细信息，请参阅库依赖项）。.prl
link_prl	启用此选项后，qmake 将处理应用程序链接到的所有库并查找其元信息（有关详细信息，请参阅库依赖项）。
no_install_prl	此选项禁用为生成的 .prl 文件生成安装规则。

注意：该选项在*构建*静态库时是必需的，而*使用*静态库时是必需的。create_prllink_prl

以下选项定义应用程序或库类型：

选择	描述
.qt	目标是Qt应用程序或库，需要Qt库和头文件。Qt库的正确包含和库路径将自动添加到项目中。这是默认定义的，可以使用变量进行微调。\l{#qt}{QT}
x11	目标是 X11 应用程序或库。正确的包含路径和库将自动添加到项目中。
测试用例	目标是自动测试。检查目标将添加到生成的生成文件中以运行测试。仅在生成生成文件时相关。
insignificant_test	自动测试的退出代码将被忽略。仅设置相关的 ifi。testcase
窗户	目标是 Win32 窗口应用程序（仅限应用）。正确的包含路径、编译器标志和库将自动添加到项目中。
安慰	目标是 Win32 控制台应用程序（仅限应用）。正确的包含路径、编译器标志和库将自动添加到项目中。请考虑使用跨平台应用程序的选项。cmdline
CMDLINE	目标是跨平台命令行应用程序。在Windows上，这意味着。在 macOS 上，这意味着。CONFIG += consoleCONFIG -= app_bundle
共享	目标是共享对象/DLL。正确的包含路径、编译器标志和库将自动添加到项目中。请注意，也可以在所有平台上使用;将创建具有目标平台（.dll 或 .so）相应后缀的共享库文件。dll
.dll	
静态的	目标是静态库（仅限库）。正确的编译器标志将自动添加到项目中。
静态库	
.plugin	目标是插件（仅限库）。这也启用了 dll。
设计师	目标是*Qt Designer*的插件。
no_lflags_merge	确保存储在变量中的库列表在使用之前不会简化为唯一值列表。LIBS
元型	为当前 project.is 创建TARGET 的全小写基名称的文件。<name>_metatypes.json<name>
qmltype	自动注册 C++ 中定义的 QML 类型。有关更多信息，请参阅从C++定义 QML 类型。另外，为当前项目创建 afile.will 是（复数，出于历史原因）ifis set，否则为TEMPLATE的值。<template>.qmltypes<template>pluginspluginqmltypesmetatypes

这些选项仅在 Windows 上定义特定功能：

选择	描述
平	使用 vcapp 模板时，这会将所有源文件放入源组，将头文件放入标头组，无论它们位于哪个目录中。关闭此选项将根据源/标头组中的文件所在的目录对文件进行分组。默认情况下处于打开状态。
embed_manifest_dll	在作为库项目的一部分创建的 DLL 中嵌入清单文件

有关嵌入清单文件选项的详细信息，请参阅平台说明。

以下选项仅在 macOS 上生效：

选择	描述
app_bundle	将可执行文件放入捆绑包中（这是默认值）。
lib_bundle	将库放入库包中。
plugin_bundle	将插件放入插件包中。Xcode 项目生成器不支持此值。

捆绑包的构建过程还受QMAKE_BUNDLE_DATA变量内容的影响。

以下选项仅在 Linux/Unix 平台上生效：

选择	描述
大文件	包括对大文件的支持。
separate_debug_info	将库的调试信息放在单独的文件中。

解析作用域时，还将检查变量。您可以为此变量分配任何内容。CONFIG

例如：

```
CONFIG += console newstuff
...
newstuff {
    SOURCES += new.cpp
    HEADERS += new.h
}
```

定义

qmake 将此变量的值添加为编译器 C 预处理器宏（-D 选项）。

例如：

```
DEFINES += USE_MY_STUFF
```

DEFINES_DEBUG

指定调试配置的预处理器定义。加载项目后，此变量的值将添加到DEFINE中。此变量通常在qmake.conf中设置，很少需要修改。

此变量在Qt 5.13.2中引入。

指定发布配置的预处理器定义。加载项目后，此变量的值将添加到DEFINE中。此变量通常在qmake.conf中设置，很少需要修改。

> **注意：**对于 MSVC mkspecs，此变量默认包含值。NDEBUG

此变量在Qt 5.13.2中引入。

DEF_FILE

> **注意：**此变量仅在使用模板时在 Windows 上使用。app

指定要包含在项目中的文件。.def

依赖路径

指定 qmake 要扫描的目录列表，以解析依赖项。当 qmake 爬网访问源代码中的头文件时，将使用此变量。#include

德斯特迪尔

指定放置目标文件的位置。

例如：

```
DESTDIR = ../../lib
```

> **注意：**支持的字符列表可能取决于使用的生成工具。特别是，括号不起作用。make

目录

指定要包含在 dist 目标中的文件列表。此功能仅受 UnixMake 规范支持。

例如：

```
DISTFILES += ../program.txt
```

DLLDESTDIR

指定将目标 dll 复制到的位置。

EXTRA_TRANSLATIONS

指定包含用户界面文本翻译成非本地语言的翻译 （.ts） 文件的列表。

与TRANSLATIONS 相反，翻译文件将仅由lrelease 处理，而不是由 lupdate 处理。EXTRA_TRANSLATIONS

您可以使用 CONFIG += lrelease 在构建过程中自动编译文件，并使用CONFIG += lrelease embed_translations使它们在Qt 资源系统中可用。

有关Qt国际化（i18n）和本地化（l10n）的更多信息，请参阅Qt语言学家手册。

形式

指定编译前要处理的UI文件（参见Qt设计器手册）。生成这些 UI 文件所需的所有依赖项、标头和源文件将自动添加到项目中。uic

例如：

```
FORMS = mydialog.ui \
    mywidget.ui \
        myconfig.ui
```

图形用户界面

指定在文件内设置的 GUID。GUID 通常是随机确定的。但是，如果需要固定的 GUID，则可以使用此变量进行设置。.vcproj

此变量仅特定于文件;否则将忽略它。.vcproj

头

定义项目的头文件。

qmake会自动检测标头中的类是否需要MOC，并将相应的依赖项和文件添加到项目中以生成和链接MOC文件。

例如：

```
HEADERS = myclass.h \
          login.h \
          mainwindow.h
```

另请参阅来源。

图标

IDL来源

此变量仅在 Windows 上用于 Visual Studio 项目生成，以将指定的文件放在"生成的文件"文件夹中。

包含路径

指定编译项目时应搜索的#include目录。

例如：

```
INCLUDEPATH = c:/msdev/include d:/stl/include
```

若要指定包含空格的路径，请使用空格中所述的方法对该路径引用。

```
win32:INCLUDEPATH += "C:/mylibs/extra headers"
unix:INCLUDEPATH += "/home/user/extra headers"
```

安装

指定在执行类似安装过程时将安装的资源列表。列表中的每个项目通常使用属性进行定义，这些属性提供有关其安装位置的信息。`make install`

例如，以下定义描述了构建目标的安装位置，并且分配将生成目标添加到要安装的现有资源列表中：`target.pathINSTALLS`

```
target.path += $$[QT_INSTALL_PLUGINS]/imageformats
INSTALLS += target
```

`INSTALLS`具有可以采用多个值的成员：`.CONFIG`

价值	描述
no_check_exist	如果未设置，qmake 会查看要安装的文件是否确实存在。如果这些文件不存在，qmake 不会创建安装规则。如果您需要安装作为构建过程的一部分生成的文件，例如 qdoc 创建的 HTML 文件，请使用此配置值。
诺斯蒂普	如果设置，将关闭典型的 Unix 条带功能，调试信息将保留在二进制文件中。
可执行	在 Unix 上，这会设置可执行标志。
no_build	当您执行 a 并且还没有项目的生成时，将首先生成该项目，然后安装该项目。如果不希望此行为，请设置此配置值以确保不会将生成目标作为依赖项添加到安装目标。`make install`
no_default_install	项目有一个顶级项目目标，当您执行 a 时，将安装所有内容。但是，如果您有一个设置了此

有关详细信息，请参阅安装文件。

此变量还用于指定将哪些附加文件部署到嵌入式设备。

JAVA_HOME

> **注意：**此变量仅对安卓目标有用。

指定用于生成项目的 JDK/OpenJDK 安装路径。

词典

指定 Lex 实现文件的列表。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

莱克斯对象

指定中间 Lex 对象文件的名称。此变量的值通常由 qmake 处理，很少需要修改。

词典来源

指定 Lex 源文件的列表。所有依赖项、头文件和源文件将自动添加到项目中，以构建这些 lex 文件。

例如：

```
LEXSOURCES = lexer.l
```

库

指定要链接到项目中的库的列表。如果使用 Unix（库）和 -L（库路径）标志，qmake 会在 Windows 上正确处理库（即将库的完整路径传递给链接器）。该库必须存在，qmake 才能找到 alib 所在的目录。`-l-l`

例如：

```
unix:LIBS += -L/usr/local/lib -lmath
win32:LIBS += c:/mylibs/math.lib
```

若要指定包含空格的路径，请使用空格中所述的方法对该路径引用。

```
win32:LIBS += "C:/mylibs/extra libs/extra.lib"
unix:LIBS += "-L/home/user/extra libs" -lextra
```

变量：LIBSno_lflags_merge

```
CONFIG += no_lflags_merge
```

LIBS_PRIVATE

指定要私下链接到项目中的库列表。此变量的行为与LIBS 相同，只是为 Unix 构建的共享库项目不会在其链接接口中公开这些依赖项。

这样做的效果是，如果项目 C 依赖于库 B，而库 B 私下依赖于库 A，但 C 也想直接使用来自 A 的符号，则需要显式链接到 A。换句话说，私有链接的库不会在构建时传递公开。

LITERAL_HASH

每当变量声明中需要文本哈希字符 （） 时，就会使用此变量，该字符可能作为文件名的一部分或在传递给某个外部应用程序的字符串中。#

例如：

```
# To include a literal hash character, use the $$LITERAL_HASH variable:
urlPieces = http://doc.qt.io/qt-5/qtextdocument.html pageCount
message($$join(urlPieces, $$LITERAL_HASH))
```

通过以这种方式使用，字符可用于构造一个 URL，以便函数打印到控制台。LITERAL_HASH#message()

生成文件

指定生成的生成文件的名称。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

MAKEFILE_GENERATOR

指定生成生成文件时要使用的生成文件生成器的名称。此变量的值通常由 qmake 在内部处理，很少需要修改。

MSVCPROJ_*

这些变量由 qmake 在内部处理，不应修改或使用。

MOC_DIR

指定应放置所有中间 moc 文件的目录。

例如：

```
win32:MOC_DIR = c:/myproject/tmp
```

OBJECTIVE_HEADERS

定义项目的 Objective-C++ 头文件。

qmake会自动检测标头中的类是否需要MOC，并将相应的依赖项和文件添加到项目中以生成和链接MOC文件。

这类似于 HEADERS 变量，但会允许生成的 moc 文件使用 Objective-C++ 编译器进行编译。

另请参阅OBJECTIVE_SOURCES。

OBJECTIVE_SOURCES

指定项目中所有 Objective-C/C++ 源文件的名称。

这个变量现在已经过时了，Objective-C/C++文件（.m和.mm）可以添加到SOURCES变量中。

另请参阅OBJECTIVE_HEADERS。

对象

此变量是从SOURCES变量自动填充的。每个源文件的扩展名被替换为.o（Unix）或.obj（Win32）。您可以将对象添加到列表中。

OBJECTS_DIR

指定应放置所有中间对象的目录。

例如：

```
unix:OBJECTS_DIR = ../myproject/tmp
win32:OBJECTS_DIR = c:/myproject/tmp
```

POST_TARGETDEPS

列出目标所依赖的库。某些后端（如 Visual Studio 和 Xcode 项目文件的生成器）不支持此变量。通常，这些构建工具在内部支持此变量，它对于显式列出依赖的静态库很有用。

此列表放在所有内置（和$$PRE_TARGETDEPS）依赖项之后。

PRE_TARGETDEPS

列出目标所依赖的库。某些后端（如 Visual Studio 和 Xcode 项目文件的生成器）不支持此变量。通常，这些构建工具在内部支持此变量，它对于显式列出依赖的静态库很有用。

PRECOMPILED_HEADER

指示用于创建预编译头文件的头文件，以提高项目的编译速度。预编译头目前仅在某些平台上受支持（Windows - 所有MSVC项目类型，Apple - Xcode，Makefile，Unix - gcc 3.3及更高版本）。

残疾人

指定指向包含当前正在分析的文件的目录的完整路径。在编写项目文件以支持影子生成时，这对于引用源代码树中的文件非常有用。

另请参阅_PRO_FILE_PWD_。

> **注意：**不要尝试覆盖此变量的值。

OUT_PWD

指定指向 qmake 放置生成的生成文件的目录的完整路径。

> **注意：**不要尝试覆盖此变量的值。

QM_FILES_RESOURCE_PREFIX

指定资源系统中的目录，其中文件将由CONFIG += embed_translations提供。`.qm`

默认值为。`:/i18n/`

QM_FILES_INSTALL_PATH

指定目标目录由CONFIG += lrelease生成的文件将安装到。如果设置了CONFIG += embed_translations，则没有任何效果。`.qm`

QML_IMPORT_PATH

此变量仅由Qt Creator使用。如果你有一个额外的模块被保留在Qt安装之外，你可以在这里指定它的路径。

有关详细信息，请参阅Qt Creator：将 QML 模块与插件一起使用。

量子路径

需要指向 QML 模块树的根目录的导入路径列表。例如，如果您有 QML 模块的自定义位置，则可以在此处指定该位置。

> **注意：**QMLPATH 的路径条目指向 QML 模块树的根目录。这是 QML 引擎理解的导入路径的概念。您可以通过环境变量将相同的路径传递给 QML 应用程序，但它们与QML_IMPORT_PATHqmake变量的预期内容不同。后者

> **注意：** QMLPATH 的内容不会自动传递到应用程序。相反，它们仅在构建时使用。特别是，qmlimportscanner 使用它们来查找可能需要标记为由应用程序导入的任何 QML 模块。

QMAKE

指定 qmake 程序本身的名称，并放置在生成的生成文件中。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKESPEC

一个系统变量，其中包含生成生成文件时使用的 qmake 配置的完整路径。此变量的值是自动计算的。

> **注意：** 不要尝试覆盖此变量的值。

QMAKE_AR_CMD

> **注意：** 此变量仅在 Unix 平台上使用。

指定创建共享库时要执行的命令。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_BUNDLE_DATA

> **注意：** 此变量仅在 macOS、iOS、tvOS 和 watchOS 上使用。

指定将与库捆绑包一起安装的数据，通常用于指定头文件的集合。

例如，以下行将添加到一个组，其中包含有关随框架提供的标头的信息：
`path/to/header_one.hpath/to/header_two.h`

```
FRAMEWORK_HEADERS.version = Versions
FRAMEWORK_HEADERS.files = path/to/header_one.h path/to/header_two.h
FRAMEWORK_HEADERS.path = Headers
QMAKE_BUNDLE_DATA += FRAMEWORK_HEADERS
```

最后一行将有关标头的信息添加到将与库捆绑包一起安装的资源集合中。

将选项添加到CONFIG变量时，将创建库捆绑包。`lib_bundle`

有关创建库捆绑包的更多信息，请参阅平台说明。

项目还可以使用此变量来捆绑应用程序翻译文件。确切的语法取决于项目使用的是 Xcode 的旧构建系统还是新的构建系统。

```
translations_en.files = $$PWD/en.lproj/InfoPlist.strings
translations_en.path = en.lproj
QMAKE_BUNDLE_DATA += translations_en
```

Xcode 将忽略 的原始位置，并且文件将放置在提供的路径下的 bundle 目录中，因此 `InfoPlist.stringsResourcestranslations_en.pathResources/en.lproj/InfoPlist.strings`

使用新的构建系统，将保留文件的相对位置，这意味着文件将被错误地放置在 `Resources/en.lproj/en.lproj/InfoPlist.strings`

为了确保正确的文件放置，项目可以将原始文件移动到不在子目录中，也可以选择不指定变量。 `translations_en.path`

```
# Approach 1
translations_en.files = $$PWD/InfoPlist.strings
translations_en.path = en.lproj

# Approach 2
translations_de.files = $$PWD/de.lproj/InfoPlist.strings

QMAKE_BUNDLE_DATA += translations_en translations_de
```

请参阅QTBUG-98417了解有关 Xcode 构建系统如何在捆绑翻译文件时更改其行为的更多详细信息。

QMAKE_BUNDLE_EXTENSION

> **注意：**此变量仅在 macOS、iOS、tvOS 和 watchOS 上使用。

指定要用于库捆绑包的扩展名。这允许使用自定义扩展而不是标准目录名称扩展创建框架。 `.framework`

例如，以下定义将生成一个带有扩展的框架： `.myframework`

```
QMAKE_BUNDLE_EXTENSION = .myframework
```

QMAKE_CC

指定在生成包含 C 源代码的项目时将使用的 C 编译器。只需指定编译器可执行文件的文件名，只要它在处理 Makefile 时位于变量中包含的路径上即可。 `PATH`

QMAKE_CFLAGS

指定用于生成项目的 C 编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。特定于调试和发

QMAKE_CFLAGS_DEBUG

指定调试版本的 C 编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_CFLAGS_RELEASE

指定发布版本的 C 编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_CFLAGS_RELEASE_WITH_DEBUGINFO

指定在其中设置的发布版本的 C 编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。`force_debug_infoCONFIG`

QMAKE_CFLAGS_SHLIB

> **注意：**此变量仅在 Unix 平台上使用。

指定用于创建共享库的编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_CFLAGS_THREAD

指定用于创建多线程应用程序的编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_CFLAGS_WARN_OFF

仅当设置了CONFIG选项时，才使用此变量。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。`warn_off`

QMAKE_CFLAGS_WARN_ON

仅当设置了CONFIG选项时，才使用此变量。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。`warn_on`

QMAKE_CLEAN

指定生成的文件（例如，按moc和uic）和要删除的对象文件的列表。`make clean`

QMAKE_CXX

指定在生成包含C++源代码的项目时将使用的C++编译器。只需指定编译器可执行文件的文件名，只要它在处理 Makefile 时位于变量中包含的路径上即可。`PATH`

指定用于生成项目的C++编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。特定于调试和发布模式的标志可以分别通过修改 和变量来调整。QMAKE_CXXFLAGS_DEBUGQMAKE_CXXFLAGS_RELEASE

QMAKE_CXXFLAGS_DEBUG

指定调试版本的C++编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_CXXFLAGS_RELEASE

指定发布版本的C++编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_CXXFLAGS_RELEASE_WITH_DEBUGINFO

指定在其中设置的发布版本的C++编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。
`force_debug_infoCONFIG`

QMAKE_CXXFLAGS_SHLIB

指定用于创建共享库的C++编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_CXXFLAGS_THREAD

指定用于创建多线程应用程序的C++编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_CXXFLAGS_WARN_OFF

指定用于禁止显示编译器警告的C++编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_CXXFLAGS_WARN_ON

指定用于生成编译器警告C++编译器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_DEVELOPMENT_TEAM

> **注意：** 此变量仅在 macOS、iOS、tvOS 和 watchOS 上使用。

用于对证书和预配配置文件进行签名的开发团队的标识符。

QMAKE_DISTCLEAN

指定要删除的文件的列表。`make distclean`

包含共享库的扩展。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

> **注意：**更改扩展的平台特定变量将覆盖此变量的内容。

QMAKE_EXTENSION_STATICLIB

包含共享静态库的扩展。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_EXT_MOC

包含用于包含的 moc 文件的扩展名。

另请参阅文件扩展名。

QMAKE_EXT_UI

包含*Qt Designer*UI文件上使用的扩展名。

另请参阅文件扩展名。

QMAKE_EXT_PRL

包含用于创建的 PRL 文件的扩展名。

另请参阅文件扩展名，库依赖项。

QMAKE_EXT_LEX

包含用于提供给 Lex 的文件的扩展名。

另请参阅文件扩展名，LEXSOURCES。

QMAKE_EXT_YACC

包含用于提供给Yacc的文件的扩展名。

另请参阅文件扩展名，YACCSOURCES。

QMAKE_EXT_OBJ

包含用于生成的对象文件的扩展名。

另请参阅文件扩展名。

包含应解释为C++源代码的文件的后缀。

另请参阅文件扩展名。

QMAKE_EXT_H

包含应解释为 C 头文件的文件的后缀。

另请参阅文件扩展名。

QMAKE_EXTRA_COMPILERS

指定其他编译器或预处理器的列表。

另请参阅添加编译器。

QMAKE_EXTRA_TARGETS

指定其他 qmake 目标的列表。

另请参阅添加自定义目标。

QMAKE_FAILED_REQUIREMENTS

包含失败要求的列表。此变量的值由 qmake 设置，无法修改。

另请参阅requires（）和REQUIRES。

QMAKE_FRAMEWORK_BUNDLE_NAME

> **注意：**此变量仅在 macOS、iOS、tvOS 和 watchOS 上使用。

在框架项目中，此变量包含要用于生成的框架的名称。

默认情况下，此变量包含与TARGET变量相同的值。

有关创建框架和库捆绑包的更多信息，请参阅创建框架。

QMAKE_FRAMEWORK_VERSION

> **注意：**此变量仅在 macOS、iOS、tvOS 和 watchOS 上使用。

对于生成目标是 macOS、iOS、tvOS 或 watchOS 框架的项目，此变量用于指定将应用于生成的框架的版本号。

默认情况下，此变量包含与VERSION变量相同的值。

有关创建框架的更多信息，请参阅创建框架。

QMAKE_HOST

提供有关运行 qmake 的主机的信息。例如，您可以从中检索主机体系结构。QMAKE_HOST.arch

钥匙	值
。拱	主机体系结构
。操作系统	主机操作系统
.cpu_count	可用 CPU 数
。名字	主计算机名称
。版本	主机操作系统版本号
.version_string	主机操作系统版本字符串

```
win32-g++:contains(QMAKE_HOST.arch, x86_64):{
    message("Host is 64bit")
    ...
}
```

QMAKE_INCDIR

指定追加到INCLUDEPATH 的系统标头路径的列表。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_INCDIR_EGL

指定在构建具有 OpenGL/ES 或 OpenVG 支持的目标时要添加到INCLUDEPATH的 EGL 头文件的位置。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_INCDIR_OPENGL

指定在构建具有 OpenGL 支持的目标时要添加到INCLUDEPATH的 OpenGL 头文件的位置。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

如果 OpenGL 实现使用 EGL（大多数 OpenGL/ES 系统），则可能还需要设置QMAKE_INCDIR_EGL。

QMAKE_INCDIR_OPENGL_ES2

此变量指定在构建具有 OpenGL ES 2 支持的目标时要添加到INCLUDEPATH的 OpenGL 头文件的位置。

此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

如果 OpenGL 实现使用 EGL（大多数 OpenGL/ES 系统），则可能还需要设置QMAKE_INCDIR_EGL。

QMAKE_INCDIR_OPENVG

如果OpenVG实现使用EGL，则可能还需要设置QMAKE_INCDIR_EGL。

QMAKE_INCDIR_X11

> **注意：**此变量仅在 Unix 平台上使用。

指定在生成 X11 目标时要添加到INCLUDEPATH的 X11 头文件路径的位置。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_INFO_PLIST

> **注意：**此变量仅在 macOS、iOS、tvOS 和 watchOS 平台上使用。

指定要包含在 macOS、iOS、tvOS 和 watchOS 应用程序捆绑包中的属性列表文件的名称。`.plist`

在文件中，您可以定义一些变量，qmake 将用相关值替换这些变量：`.plist`

占位符	影响
`${PRODUCT_BUNDLE_IDENTIFIER}`, `@BUNDLEIDENTIFIER@`	扩展到目标分发包的分发包标识符字符串，例如：。通过连接 QMAKE_TARGET_BUNDLE_PREFIX 和 QMAKE_BUNDLE 的值来确定，用句号 () 分隔。`com.example.myapp.`
`${EXECUTABLE_NAME}`,, `@EXECUTABLE@@LIBRARY@`	等效于 QMAKE_APPLICATION_BUNDLE_NAME、QMAKE_PLUGIN_BUNDLE_NAME 或 QMAKE_FRAMEWORK_BUNDLE_NAME（取决于要创建的目标的类型）的值，如果未设置前面的任何值，则为 TARGET。
`${ASSETCATALOG_COMPILER_APPICON_NAME}`, `@ICON@`	展开为 ICON 的值。
`${QMAKE_PKGINFO_TYPEINFO}`, `@TYPEINFO@`	展开为QMAKE_PKGINFO_TYPEINFO值。
`${QMAKE_FULL_VERSION}`, `@FULL_VERSION@`	扩展到由三个版本组件表示的 VERSION 值。
`${QMAKE_SHORT_VERSION}`, `@SHORT_VERSION@`	扩展到用两个版本组件表示的 VERSION 值。
`${MACOSX_DEPLOYMENT_TARGET}`	扩展到 QMAKE_MACOSX_DEPLOYMENT_TARGET 的值。
`${IPHONEOS_DEPLOYMENT_TARGET}`	扩展到 QMAKE_IPHONEOS_DEPLOYMENT_TARGET 的值。
`${TVOS_DEPLOYMENT_TARGET}`	扩展到 QMAKE_TVOS_DEPLOYMENT_TARGET 的值。
`${WATCHOS_DEPLOYMENT_TARGET}`	扩展到 QMAKE_WATCHOS_DEPLOYMENT_TARGET 的值。
`${IOS_LAUNCH_SCREEN}`	扩展到 QMAKE_IOS_LAUNCH_SCREEN 的值。

> **注意：**使用 Xcode 生成器时，上述样式的占位符直接由 Xcode 构建系统替换，不由 qmake 处理。样式占位符仅适用于 qmake Makefile 生成器，而不适用于 Xcode 生成器。`${var}@var@`

> **注意：**大多数时候，默认值就足够了。`Info.plist`

QMAKE_IOS_DEPLOYMENT_TARGET

> **注意：**此变量仅在 iOS 平台上使用。

指定应用程序支持的 iOS 的最低硬版本。

有关详细信息，请参阅表达支持的 iOS 版本。

QMAKE_IOS_LAUNCH_SCREEN

> **注意：**此变量仅在 iOS 平台上使用。

指定应用程序使用的启动屏幕。如果未设置，则使用默认启动屏幕。

QMAKE_LFLAGS

指定传递给链接器的一组常规标志。如果需要更改用于特定平台或项目类型的标志，请使用专用变量之一而不是此变量。

QMAKE_LFLAGS_CONSOLE

> **注意：**此变量仅在 Windows 上使用。

指定用于生成控制台程序的链接器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LFLAGS_DEBUG

指定调试版本的链接器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LFLAGS_PLUGIN

指定用于生成插件的链接器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LFLAGS_RPATH

> **注意：**此变量仅在 Unix 平台上使用。

此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LFLAGS_REL_RPATH

指定在QMAKE_RPATHDIR中启用相对路径所需的链接器标志。

此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_REL_RPATH_BASE

指定动态链接器将字符串理解为引用可执行文件或库的位置。

此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LFLAGS_RPATHLINK

指定使用QMAKE_RPATHLINKDIR中的值所需的链接器标志。

此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LFLAGS_RELEASE

指定发布版本的链接器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LFLAGS_RELEASE_WITH_DEBUGINFO

指定在其中设置的发布版本的链接器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。
`force_debug_infoCONFIG`

QMAKE_LFLAGS_APP

指定用于生成应用程序的链接器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LFLAGS_SHLIB

指定用于生成共享库的链接器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LFLAGS_SONAME

指定用于设置共享对象（如 .so 或 .dll）名称的链接器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LFLAGS_THREAD

QMAKE_LFLAGS_WINDOWS

> **注意：**此变量仅在 Windows 上使用。

指定用于生成 Windows GUI 项目（即非控制台应用程序）的链接器标志。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LIBDIR

指定所有项目的库搜索路径列表。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

若要在项目文件中指定其他搜索路径，请改用这样的LIBS：

```
LIBS += -L/path/to/libraries
```

QMAKE_LIBDIR_POST

指定所有项目的系统库搜索路径列表。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LIBDIR_FLAGS

> **注意：**此变量仅在 Unix 平台上使用。

指定前缀为 -L 的所有库目录的位置。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LIBDIR_EGL

指定 EGL 库目录的位置，当 EGL 与 OpenGL/ES 或 OpenVG 一起使用时。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LIBDIR_OPENGL

指定 OpenGL 库目录的位置。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

如果 OpenGL 实现使用 EGL（大多数 OpenGL/ES 系统），则可能还需要设置QMAKE_LIBDIR_EGL。

QMAKE_LIBDIR_OPENVG

指定 OpenVG 库目录的位置。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

如果OpenVG实现使用EGL，则可能还需要设置QMAKE_LIBDIR_EGL。

> **注意：**此变量仅在 Unix 平台上使用。

指定 X11 库目录的位置。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LIBS

指定每个项目需要链接的其他库。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

若要在项目文件中指定库，请改用LIBS。

QMAKE_LIBS_PRIVATE

指定每个项目需要链接的其他专用库。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

若要在库项目文件中指定专用库，请改用LIBS_PRIVATE。

QMAKE_LIBS_EGL

指定使用 OpenGL/ES 或 OpenVG 构建 Qt 时的所有 EGL 库。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。通常的值是。`-lEGL`

QMAKE_LIBS_OPENGL

指定所有 OpenGL 库。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

如果 OpenGL 实现使用 EGL（大多数 OpenGL/ES 系统），则可能还需要设置QMAKE_LIBS_EGL。

QMAKE_LIBS_OPENGL_ES1， QMAKE_LIBS_OPENGL_ES2

这些变量指定了 OpenGL ES 1 和 OpenGL ES 2 的所有 OpenGL 库。

这些变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

如果 OpenGL 实现使用 EGL（大多数 OpenGL/ES 系统），则可能还需要设置QMAKE_LIBS_EGL。

QMAKE_LIBS_OPENVG

指定所有 OpenVG 库。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。通常的值是。`-lOpenVG`

一些OpenVG引擎是在OpenGL之上实现的。这将在配置时检测到，QMAKE_LIBS_OPENGL将被隐式添加到 OpenVG 库链接的任何位置QMAKE_LIBS_OPENVG。

如果 OpenVG 实现使用 EGL，则可能还需要设置QMAKE_LIBS_EGL。

QMAKE_LIBS_THREAD

> **注意：**此变量仅在 Unix 平台上使用。

指定在生成多线程目标时需要链接的所有库。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LIBS_X11

> **注意：**此变量仅在 Unix 平台上使用。

指定所有 X11 库。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LIB_FLAG

如果指定了模板，则此变量不为空。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。`lib`

QMAKE_LINK

指定在生成基于应用程序的项目时将使用的链接器。只需指定链接器可执行文件的文件名，只要它在处理 Makefile 时位于变量中包含的路径上即可。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。PATH

QMAKE_LINK_SHLIB_CMD

指定创建共享库时要执行的命令。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LN_SHLIB

指定创建指向共享库的链接时要执行的命令。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_LRELEASE_FLAGS

通过CONFIG += lrelease 启用时传递给lrelease的其他选项列表。

QMAKE_OBJECTIVE_CFLAGS

指定用于生成项目的目标 C/C++ 编译器标志。除了QMAKE_CFLAGS和QMAKE_CXXFLAGS之外，还使用这些标志。

QMAKE_POST_LINK

指定将目标链接在一起后要执行的命令。此变量通常为空，因此不执行任何内容。

> **注意：**此变量对 Xcode 项目没有影响。

指定在将目标链接在一起之前要执行的命令。此变量通常为空，因此不执行任何内容。

> **注意：**此变量对 Xcode 项目没有影响。

QMAKE_PROJECT_NAME

> **注意：**此变量仅用于 Visual Studio 项目文件。

在为 IDE 生成项目文件时确定项目的名称。默认值为目标名称。此变量的值通常由 qmake 处理，很少需要修改。

QMAKE_PROVISIONING_PROFILE

> **注意：**此变量仅在 macOS、iOS、tvOS 和 watchOS 上使用。

有效预配配置文件的 UUID。与QMAKE_DEVELOPMENT_TEAM结合使用以指定预配配置文件。

> **注意：**指定预配配置文件将禁用自动托管签名。

QMAKE_MAC_SDK

此变量在 macOS 上构建通用二进制文件时使用。

QMAKE_MACOSX_DEPLOYMENT_TARGET

> **注意：**此变量仅在 macOS 平台上使用。

指定应用程序支持的 macOS 的最低硬版本。

有关更多信息，请参阅macOS 支持的版本。

QMAKE_MAKEFILE

指定要创建的生成文件的名称。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_QMAKE

包含 qmake 可执行文件的绝对路径。

> **注意：**不要尝试覆盖此变量的值

QMAKE_RESOURCE_FLAGS

此变量用于自定义在使用它的每个生成规则中传递给资源编译器的选项列表。例如，以下行确保每次调用时都对特定值使用 theand选项：-threshold-compressrcc

```
QMAKE_RESOURCE_FLAGS += -threshold 0 -compress 9
```

QMAKE_RPATHDIR

> **注意：**此变量仅在 Unix 平台上使用。

指定在链接时添加到可执行文件的库路径列表，以便在运行时优先搜索这些路径。

指定相对路径时，qmake 会将它们修改为动态链接器理解为相对于引用可执行文件或库的位置的形式。这仅受某些平台（目前基于 Linux 和 Darwin 的平台）的支持，并且可以通过检查是否设置QMAKE_REL_RPATH_BASE来检测。

QMAKE_RPATHLINKDIR

指定静态链接器用于搜索共享库的隐式依赖项的库路径列表。有关详细信息，请参阅手册页。ld(1)

QMAKE_RUN_CC

指定生成对象所需的单个规则。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_RUN_CC_IMP

指定生成对象所需的单个规则。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_RUN_CXX

指定生成对象所需的单个规则。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_RUN_CXX_IMP

指定生成对象所需的单个规则。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_SONAME_PREFIX

如果已定义，则此变量的值将用作要附加到构建的共享库标识符前面的路径。这是动态链接器稍后将用于引用库的标识符。通常，此引用可以是库名称或完整的库路径。在 macOS、iOS、tvOS 和 watchOS 上，可以使用以下占位符相对指定路径：SONAMESONAME

@executable_path	展开到当前进程可执行文件位置。
@loader_path	展开到引用的可执行文件或库位置。

在大多数情况下，使用是足够的，建议：@rpath

```
# <project root>/project.pro
QMAKE_SONAME_PREFIX = @rpath
```

但是，也可以使用不同的占位符或绝对路径指定前缀，例如以下之一：

```
# <project root>/project.pro
QMAKE_SONAME_PREFIX = @executable_path/../Frameworks
QMAKE_SONAME_PREFIX = @loader_path/Frameworks
QMAKE_SONAME_PREFIX = /Library/Frameworks
```

有关详细信息，请参阅有关动态库安装名称的dyld文档。

QMAKE_TARGET

指定项目目标的名称。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

QMAKE_TARGET_COMPANY

仅限窗口。指定项目目标的公司;这在适用的情况下用于将公司名称放在应用程序的属性中。仅当设置了VERSION或RC_ICONS变量且未设置RC_FILE和RES_FILE变量时，才使用此方法。

QMAKE_TARGET_DESCRIPTION

仅限窗口。指定项目目标的说明;这在适用的情况下用于将说明放在应用程序的属性中。仅当设置了VERSION或RC_ICONS变量且未设置RC_FILE和RES_FILE变量时，才使用此方法。

QMAKE_TARGET_COPYRIGHT

仅限窗口。指定项目目标的版权信息;这在适用的情况下用于将版权信息放在应用程序的属性中。仅当设置了VERSION或RC_ICONS变量且未设置RC_FILE和RES_FILE变量时，才使用此方法。

QMAKE_TARGET_PRODUCT

仅限窗口。指定项目目标的产品;这在适用的情况下用于将产品放入应用程序的属性中。仅当设置了VERSION或RC_ICONS变量且未设置RC_FILE和RES_FILE变量时，才使用此方法。

仅限窗口。指定项目目标的原始文件名;这在适用的情况下用于将原始文件名放在应用程序的属性中。仅当设置了VERSION或RC_ICONS变量且未设置RC_FILE和RES_FILE变量时，才使用此方法。

QMAKE_TARGET_INTERNALNAME

仅限窗口。指定项目目标的内部名称;这在适用的情况下用于将内部名称放在应用程序的属性中。仅当设置了VERSION或RC_ICONS变量且未设置RC_FILE和RES_FILE变量时，才使用此方法。

QMAKE_TARGET_COMMENTS

仅限窗口。指定项目目标的注释;这在适用的情况下用于将注释放在应用程序的属性中。仅当设置了VERSION或RC_ICONS变量且未设置RC_FILE和RES_FILE变量时，才使用此方法。

QMAKE_TARGET_TRADEMARKS

仅限窗口。指定项目目标的商标信息;这在适用的情况下用于将商标信息放入应用程序的属性中。仅当设置了VERSION或RC_ICONS变量且未设置RC_FILE和RES_FILE变量时，才使用此方法。

QMAKE_MANIFEST

仅限窗口。指定项目目标的清单文件。仅当未设置RC_FILE和RES_FILE变量时，才使用此方法。不要忘记从CONFIG 变量中删除embed_manifest_exe和embed_manifest_dll，否则它将与编译器生成的冲突。

QMAKE_TVOS_DEPLOYMENT_TARGET

> **注意：**此变量仅在 tvOS 平台上使用。

指定应用程序支持的 tvOS 的最低硬版本。

有关详细信息，请参阅表达支持的 iOS 版本。

QMAKE_UIC_FLAGS

此变量用于自定义在使用它的每个生成规则中传递给用户界面编译器的选项列表。

QMAKE_WATCHOS_DEPLOYMENT_TARGET

> **注意：**此变量仅在 watchOS 平台上使用。

指定应用程序支持的 watchOS 的最低硬版本。

有关详细信息，请参阅表达支持的 iOS 版本。

指定要用于自动生成的 QML 类型注册的主要版本。有关更多信息，请参阅从C++定义 QML 类型。

QML_IMPORT_MINOR_VERSION

自动注册 C++ 中定义的 QML 类型时，请使用此次要版本注册模块的其他版本。通常，要注册的次要版本是从元对象推断出来的。

如果元对象未更改，并且您仍希望导入具有较新次要版本号的 QML 模块，则可以使用此变量。例如，元对象处于水平状态，但您希望将模块导入为。MyModule1.11.3

QML_IMPORT_VERSION

将QML_IMPORT_MAJOR_VERSION和QML_IMPORT_MINOR_VERSION指定为厌恶字符串。<major>.<minor>

QML_IMPORT_NAME

指定要用于自动生成的 QML 类型注册的模块名称。有关更多信息，请参阅从C++定义 QML 类型。

QML_FOREIGN_METATYPES

指定生成 qmltypes 文件时要考虑的具有元类型的其他 JSON 文件。当外部库提供直接公开给 QML 的类型或作为基类型或其他类型的属性时，请使用此选项。Qt类型将被自动考虑，不必在此处添加。

.QT

指定项目使用的Qt 模块。有关要为每个模块添加的值，请参阅模块文档。

在C++实现级别，使用 Qt 模块使其标头可供包含，并使其链接到二进制文件。

默认情况下，containsand，确保无需进一步配置即可构建标准 GUI 应用程序。QTcoregui

如果要构建一个*没有*Qt GUI模块的项目，则需要使用"-="运算符排除值。以下行将导致构建一个最小的Qt项目：gui

```
QT -= gui # Only the core module is used.
```

如果您的项目是*Qt*Designer插件，请使用value指定项目将构建为库，但具有对*Qt Designer*的特定插件支持。有关更多信息，请参阅构建和安装插件。uiplugin

QTPLUGIN

指定要与应用程序链接的静态Qt插件的名称列表，以便它们可用作内置资源。

qmake会自动添加使用的Qt模块通常需要的插件（请参阅）。默认设置已调整为最佳的开箱即用体验。有关可用插件的列表以及覆盖自动链接的方法，请参阅静态插件。QT

QT_VERSION

包含当前版本的 Qt。

QT_MAJOR_VERSION

包含 Qt 的当前主要版本。

QT_MINOR_VERSION

包含 Qt 的当前次要版本。

QT_PATCH_VERSION

包含 Qt 的当前补丁版本。

RC_FILE

仅限窗口。指定目标的 Windows 资源文件 （.rc） 的名称。请参阅添加 Windows 资源文件。

RC_CODEPAGE

仅限窗口。指定应在生成的 .rc 文件中指定的代码页。仅当设置了VERSION或RC_ICONS变量且未设置RC_FILE和RES_FILE变量时，才使用此方法。

RC_DEFINES

仅限窗口。qmake 将此变量的值添加为 RC 预处理器宏（/d 选项）。如果未设置此变量，则改用DEFINE变量。

```
RC_DEFINES += USE_MY_STUFF
```

RC_ICONS

仅限窗口。指定应包含在生成的 .rc 文件中的图标。仅当未设置RC_FILE和RES_FILE变量时，才使用此选项。有关生成 .rc 文件的更多详细信息，请参阅平台说明。

RC_LANG

仅限窗口。指定应在生成的 .rc 文件中指定的语言。仅当设置了VERSION或RC_ICONS变量且未设置RC_FILE和RES_FILE变量时，才使用此方法。

指定传递给 Windows 资源编译器的包含路径。

RCC_DIR

指定 Qt 资源编译器输出文件的目录。

例如：

```
unix:RCC_DIR = ../myproject/resources
win32:RCC_DIR = c:/myproject/resources
```

需要

指定作为条件计算的值的列表。如果任何条件为假，qmake 在构建时会跳过此项目（及其SUBDIRS）。

> **注意：**如果您想在构建时跳过项目或子项目，我们建议改用requires（） 函数。

资源

指定目标的资源集合文件 （qrc） 的名称。有关资源集合文件的更多信息，请参阅Qt 资源系统。

RES_FILE

仅限窗口。指定此目标的 Windows 资源编译器输出文件的名称。请参阅RC_FILE和添加 Windows 资源文件。

此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。

来源

指定项目中所有源文件的名称。

例如：

```
SOURCES = myclass.cpp \
          login.cpp \
          mainwindow.cpp
```

另请参阅标头。

子目录

建议每个子目录中的项目文件与子目录本身具有相同的基名称，因为这样可以省略文件名。例如，如果调用子目录，则应调用该目录中的项目文件。myappmyapp.pro

或者，可以在任何目录中指定 .pro 文件的相对路径。强烈建议您仅指定当前项目的父目录或其子目录中的路径。

例如：

```
SUBDIRS = kernel \
          tools \
          myapp
```

如果需要确保按特定顺序构建子目录，请在相关元素上使用主题符号。 .dependsSUBDIRS

例如：

```
SUBDIRS += my_executable my_library tests doc
my_executable.depends = my_library
tests.depends = my_executable
```

上面的配置确保之前构建和之前构建。但是，可以与其他子目录并行构建，从而加快构建过程。
my_librarymy_executablemy_executabletestsdoc

> **注意：**可以列出多个依赖项，它们都将在依赖于它们的目标之前构建。

> **注意：**不建议使用CONFIG += 排序，因为它会减慢多核构建的速度。与上面显示的示例不同，所有生成都将按顺序进行，即使它们没有依赖项。

除了定义构建顺序之外，还可以修改默认行为，方法是为元素提供额外的修饰符。支持的修饰符包括：
SUBDIRSSUBDIRS

修饰语	影响
.subdir	使用指定的子目录而不是值。SUBDIRS
。文件	显式指定子项目文件。不能与修饰符一起使用。pro.subdir
。取决于	此子项目依赖于指定的子项目。
.makefile	子项目的生成文件。仅在使用生成文件的平台上可用。
。目标	用于与此子项目相关的生成文件目标的基本字符串。仅在使用生成文件的平台上可用。

例如，定义两个子目录，这两个子目录都位于与值不同的目录中，并且其中一个子目录必须在另一个子目录之前构建：SUBDIRS

```
SUBDIRS += my_executable my_library
my_executable.subdir = app
```

目标

指定目标文件的名称。默认情况下包含项目文件的基名称。

例如：

```
TEMPLATE = app
TARGET = myapp
SOURCES = main.cpp
```

上面的项目文件将生成一个可执行的名称don unix andon Windows。`myappmyapp.exe`

TARGET_EXT

指定扩展名。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。`TARGET`

TARGET_x

指定具有主版本号的扩展名。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。`TARGET`

TARGET_x.y.z

使用版本号指定扩展名。此变量的值通常由 qmake 或qmake.conf处理，很少需要修改。`TARGET`

模板

指定生成项目时要使用的模板的名称。允许的值为：

选择	描述
.app	创建用于生成应用程序的生成文件（默认值）。有关详细信息，请参阅构建应用程序。
自由	创建用于构建库的生成文件。有关详细信息，请参阅构建库。
子目录	创建用于在子目录中生成目标的生成文件。子目录是使用SUBDIRS变量指定的。
辅助	创建一个不构建任何内容的生成文件。如果不需要调用编译器来创建目标，请使用此选项;例如，因为您的项目是用解释型语言编写的。 **注意：**此模板类型仅适用于基于生成文件的生成器。特别是，它不适用于vcxproj和Xcode生成器。
VC普	仅限窗口。为 Visual Studio 创建一个应用程序项目。有关详细信息，请参阅创建 Visual Studio 项目文件。

例如：

```
TEMPLATE = lib
SOURCES = main.cpp
TARGET = mylib
```

通过使用命令行选项指定新的模板类型，可以覆盖模板。这将在处理 .pro 文件*后*覆盖模板类型。对于使用模板类型来确定项目生成方式的 .pro 文件，必须在命令行上声明 TEMPLATE，而不是使用该选项。`-t-t`

翻译

指定包含用户界面文本翻译成非本地语言的翻译 （.ts） 文件的列表。

翻译文件将由lrelease和使用lupdate工具处理。如果只想处理文件，请使用EXTRA_TRANSLATIONS。`TRANSLATIONSlrelease`

您可以使用 CONFIG += lrelease 在构建过程中自动编译文件，并使用CONFIG += lrelease embed_translations使它们在Qt 资源系统中可用。

有关Qt国际化（i18n）和本地化（l10n）的更多信息，请参阅Qt语言学家手册。

UI_DIR

指定应放置 uic 中的所有中间文件的目录。

例如：

```
unix:UI_DIR = ../myproject/ui
win32:UI_DIR = c:/myproject/ui
```

版本

如果指定了模板，则指定应用程序的版本号，如果指定了模板，则指定库的版本号。`applib`

在 Windows 上，如果未设置RC_FILE和RES_FILE变量，则会触发 .rc 文件的自动生成。生成的 .rc 文件将包含 FILEVERSION 和 PRODUCTVERSION 条目，其中包含主要、次要、补丁级别和内部版本号。每个数字必须在 0 到 65535 的范围内。有关生成 .rc 文件的更多详细信息，请参阅平台说明。

例如：

```
win32:VERSION = 1.2.3.4 # major.minor.patch.build
else:VERSION = 1.2.3    # major.minor.patch
```

VERSION_PE_HEADER

要版本。如果未设置VERSION_PE_HEADER，它将从VERSION（如果设置）回退到主要版本和次要版本。

```
VERSION_PE_HEADER = 1.2
```

VER_MAJ

指定库的主版本号（如果指定了模板）。lib

VER_MIN

指定库的次要版本号（如果指定了模板）。lib

VER_PAT

指定库的修补程序版本号（如果指定了模板）。lib

VPATH

告诉 qmake 在哪里搜索它无法打开的文件。例如，如果 qmake 查找并找到无法打开的条目，它会查看整个 VPATH 列表以查看是否可以自行找到该文件。SOURCES

另请参阅依赖路径。

WINDOWS_TARGET_PLATFORM_VERSION

指定目标窗口版本;这对应于 Tagin VCXPROJ 文件。WindowsTargetPlatformVersion

在桌面 Windows 上，默认值是环境变量的值。WindowsSDKVersion

WINDOWS_TARGET_PLATFORM_MIN_VERSION

指定 Windows 目标平台的最低版本;这对应于 Tagin VCXPROJ 文件。WindowsTargetPlatformMinVersion

默认为。WINDOWS_TARGET_PLATFORM_VERSION

亚克来源

指定要包含在项目中的 Yacc 源文件的列表。所有依赖项、标头和源文件将自动包含在项目中。

例如：

```
YACCSOURCES = moc.y
```

包含正在使用的项目文件的路径。

例如，以下行导致将项目文件的位置写入控制台：

```
message($$_PRO_FILE_)
```

注意：不要尝试覆盖此变量的值。

_PRO_FILE_PWD_

包含包含正在使用的项目文件的目录的路径。

例如，以下行会导致将包含项目文件的目录的位置写入控制台：

```
message($$_PRO_FILE_PWD_)
```

注意：不要尝试覆盖此变量的值。

另请参阅QQmlEngine：：addImportPath（）。

‹ 参考 替换函数 ›

Topics ›

支持

支持服务
专业服务
合作 伙伴
训练

对于客户

支持中心
下载
Qt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**