搜索

开始使用 qmake

本教程教你qmake的基础知识。本手册中的其他主题包含有关使用 qmake 的更多详细信息。

从简单开始

假设您刚刚完成了应用程序的基本实现，并且您已经创建了以下文件：

- 你好.cpp
- 你好，H
- 主.cpp

您可以在Qt发行版的目录中找到这些文件。关于应用程序的设置，您唯一知道的另一件事是它是用Qt编写的。首先，使用您最喜欢的纯文本编辑器，创建一个名为 in 的文件。您需要做的第一件事是添加行，告诉 qmake 有关开发项目中的源文件和头文件的信息。
`examples/qmake/tutorialhello.proexamples/qmake/tutorial`

我们将首先将源文件添加到项目文件中。为此，您需要使用SOURCES变量。只需开始一行新行，并在它之后加上 hello.cpp。你应该有这样的东西：SOURCES +=

```
SOURCES += hello.cpp
```

我们对项目中的每个源文件重复此操作，直到最终得到以下内容：

```
SOURCES += hello.cpp
SOURCES += main.cpp
```

如果您更喜欢使用类似 Make 的语法，一次性列出所有文件，您可以使用换行符转义，如下所示：

```
SOURCES = hello.cpp \
          main.cpp
```

变量名称是HEADERS。

完成此操作后，项目文件应如下所示：

```
HEADERS += hello.h
SOURCES += hello.cpp
SOURCES += main.cpp
```

目标名称是自动设置的。它与项目文件名相同，但具有适合平台的后缀。例如，如果调用项目文件，则目标将是Windows和Unix。如果要使用其他名称，可以在项目文件中设置它：hello.prohello.exehello

```
TARGET = helloworld
```

完成的项目文件应如下所示：

```
HEADERS += hello.h
SOURCES += hello.cpp
SOURCES += main.cpp
```

现在，您可以使用 qmake 为您的应用程序生成生成文件。在命令行的项目目录中，键入以下内容：

> **注意：**如果你通过包管理器安装了Qt，二进制文件可能是。qmake6

然后类型或取决于您使用的编译器。makenmake

对于Visual Studio用户，qmake还可以生成Visual Studio项目文件。例如：

```
qmake -tp vc hello.pro
```

使应用程序可调试

应用程序的发布版本不包含任何调试符号或其他调试信息。在开发过程中，生成具有相关信息的应用程序的调试版本非常有用。这可以通过添加到项目文件中的CONFIG变量轻松实现。debug

例如：

```
SOURCES += main.cpp
```

像以前一样使用 qmake 生成生成文件。现在，在调试环境中运行应用程序时，您将获得有关应用程序的有用信息。

添加特定于平台的源文件

经过几个小时的编码后，您可能已经开始了解应用程序的特定于平台的部分，并决定将依赖于平台的代码分开。因此，您现在有两个新文件要包含在项目文件中：and。我们不能只是将它们添加到变量中，因为这会将两个文件都放在 Makefile 中。因此，我们在这里需要做的是使用一个范围，该范围将根据我们正在构建的平台进行处理。`hellowin.cpphellounix.cppSOURCES`

为 Windows 添加与平台相关的文件的简单作用域如下所示：

```
win32 {
    SOURCES += hellowin.cpp
}
```

在为 Windows 构建时，qmake 会添加到源文件列表中。在为任何其他平台构建时，qmake 会忽略它。现在剩下要做的就是为 Unix 特定的文件创建一个范围。`hellowin.cpp`

完成此操作后，项目文件应如下所示：

```
CONFIG += debug
HEADERS += hello.h
SOURCES += hello.cpp
SOURCES += main.cpp
win32 {
    SOURCES += hellowin.cpp
}
unix {
    SOURCES += hellounix.cpp
}
```

像以前一样使用 qmake 生成生成文件。

如果文件不存在，则停止 qmake

如果某个文件不存在，您可能不希望创建生成文件。我们可以通过使用exists（）函数检查文件是否存在。我们可以通过使用error（）函数来阻止 qmake 处理。其工作方式与作用域相同。只需将范围条件替换为函数即可。对名为 main.cpp 的文件的检查如下所示：

```
!exists( main.cpp ) {
    error( "No main.cpp file found" )
```

符号用于否定测试。也就是说，如果文件存在，则为 true，如果文件不存在，则为 true。!exists(main.cpp)!exists(main.cpp)

```
CONFIG += debug
HEADERS += hello.h
SOURCES += hello.cpp
SOURCES += main.cpp
win32 {
    SOURCES += hellowin.cpp
}
unix {
    SOURCES += hellounix.cpp
}
!exists( main.cpp ) {
    error( "No main.cpp file found" )
}
```

像以前一样使用 qmake 生成生成文件。如果您暂时重命名，您将看到该消息，qmake 将停止处理。main.cpp

检查多个条件

假设您使用 Windows，并且希望在命令行上运行应用程序时能够看到语句输出。若要查看输出，必须使用适当的控制台设置生成应用程序。我们可以轻松地将此设置包含在Windows上的Makefile中。但是，假设我们只想在Windows上运行时*添加*该行，并且何时已经在线。这需要使用两个嵌套作用域。首先创建一个作用域，然后在其中创建另一个作用域。将要处理的设置放在第二个作用域内，如下所示：
qDebug()consoleCONFIGCONFIGdebugCONFIG

```
win32 {
    debug {
        CONFIG += console
    }
}
```

嵌套作用域可以使用冒号连接在一起，因此最终项目文件如下所示：

```
CONFIG += debug
HEADERS += hello.h
SOURCES += hello.cpp
SOURCES += main.cpp
win32 {
    SOURCES += hellowin.cpp
}
unix {
    SOURCES += hellounix.cpp
}
!exists( main.cpp ) {
```

```
    CONFIG += console
}
```

就是这样！您现在已完成 qmake 教程，并准备好为开发项目编写项目文件。

‹ 概述 创建项目文件 ›