<http://blog.csdn.net/dbzhang800/article/details/6348432>

尽管每次和cmake对比起来，我们总是说 qmake 简单、功能少。但是qmake仍然是一个非常复杂的东西，我想大多人应该和我一样吧：

* 不是太清楚CONFIG等变量到底如何起作用的
* 用过的qmake内置变量和函数不超过20个
* 看Qt Creator源码或者QtSolution等库中自己使用的 \*.pro 文件就想看天书一样

本文只能抓住一条线，简单介绍一下 \*.pro、\*.pri、\*.prf、\*.prl等四种文件：干嘛用的，如何用的

## \*.pro

qmake 的工程(**pro**ject)文件，这个大家肯定都非常熟悉了。那我就不费话了，上例子：

这是一个典型的Qt示例程序的.pro文件(propriprfprl.pro)：

TEMPLATE = app

CONFIG += QT

QT += core gui

TARGET = propriprfprl

SOURCES += main.cpp/

widget.cpp

HEADERS += widget.h

FORMS += widget.ui

* 前面3行是qmake的默认值，我们都可以省略
* TARGET 这行指定工程名，我们也可以省略

## \*.pri

i 是什么东西？包含(**i**nclude)的首字母。类似于C、C++中的头文件吧，反正就是我们可以吧 \*.pro 文件内的一部分单独放到一个 \*.pri 文件内，然后包含进来。

接前面的例子，我们将源文件的设置独立出来，放到propriprfprl.pri文件内：

SOURCES += main.cpp/

widget.cpp

HEADERS += widget.h

FORMS += widget.ui

这时，我们的 propriprfprl.pro 文件就可以简化为：

TEMPLATE = app

CONFIG += QT

QT += core gui

TARGET = propriprfprl

include(propriprfprl.pri)

* 这有什么用呢？对我们这个例子来说，确实没什么用，反而多了一个文件，更麻烦了。
* 可是，如果一个大点的项目，含有多个\*.pro文件呢，这些pro需要有些共同的设置或需要的文件，这时就很有必要了。

## \*.prf

f又是神马东东？特性(**f**eature)的首字符

* 和pri文件类似，该文件也是要被包含进pro文件的
* 只是：它更隐蔽
* 你经常和它打交道，可能却一直视而不见

我们这个例子中其实已经用到了，这就是

CONFIG+=QT

当我们在CONFIG中指定一个东西时，qmake就会尝试去加载相应的feature文件：

* Qt安装目录下的 mkspecs/features/qt.prf
* features 文件的文件名必须小写
* qmake 去哪些目录下搜索features文件呢？
  + manual中有介绍，此处略
  + 暂时只知道前面提到的 $$QTDIR/mkspecs/features 就可以了

写一个自己的features文件：propriprfprl.prf

win32:CONFIG += console

* 为win32的程序添加控制台，有点多次一举哈。
* 将该文件放置到我们前面提到的目录中

然后在pro文件内添加

CONFIG += propriprfprl

看到和 CONFIG += console 同样的效果了吧？

注：我们也可以使用 load命令来加载prf文件，比如前面的命令可以认为等价于

load(propriprfprl)

## \*.prl

l 这个东西容易理解，链接(**l**ink)的首字符。主要和生成与使用静态库密切相关(动态库也可以有该文件，去Qt安装目录下的lib目录下看看即可)。

* 生成静态库时，我们需要使用下列配置(进而生成和库文件同名的 \*.prl 文件)

CONFIG += create\_prl

* 当工程的TEMPLATE为app时，会自动添加如下指令(找库文件的时候，会尝试找相应的 \*.prl 文件)

CONFIG += link\_pri

那么该文件有什么用处呢？举一个大家可能熟悉的例子QextSerialPort1.2这个库(windows下的情况)：

* 编译时，需要 setupapi.lib advapi32.lib user32.lib 这几个库文件
* 编译成静态库以后，它本身是不包含这3个库文件信息的
* 于是，当我们使用这个 QextSerialPort 静态库，还是需要指定 这几个库文件

如果有prl文件呢，该文件就会包含依赖信息了，我们看一下：

QMAKE\_PRL\_BUILD\_DIR = E:/dbzhang800-qextserialport/buildlib

QMAKE\_PRO\_INPUT = buildlib.pro

QMAKE\_PRL\_TARGET = qextserialport-1.2

QMAKE\_PRL\_CONFIG = include\_source\_dir incredibuild\_xge lex yacc warn\_on uic resources incremental\_off windows release ReleaseBuild Release build\_pass qt warn\_on release incremental flat link\_prl precompile\_header autogen\_precompile\_source copy\_dir\_files debug\_and\_release debug\_and\_release\_target embed\_manifest\_dll embed\_manifest\_exe stl exceptions rtti mmx 3dnow sse sse2 release ReleaseBuild Release build\_pass qt qextserialport-buildlib create\_prl qextserialport-uselib qextserialport-static static debug\_and\_release build\_all release ReleaseBuild Release build\_pass no\_autoqmake staticlib static moc thread

QMAKE\_PRL\_LIBS = setupapi.lib advapi32.lib user32.lib d://Qt//4.7.0//lib//QtGui4.lib d://Qt//4.7.0//lib//QtCore4.lib

## 参考

* qmake manual:qmake-advanced-usage.html
* qmake manual:qmake-variable-reference.html
* <http://www.qtcentre.org/wiki/index.php?title=Undocumented_qmake>