**关于Qt中使用的工程文件(.pro文件)**

**(2009-11-02 14:57:11)**

**摘自：**[**http://blog.chinaunix.net/u1/37292/showart\_287681.html**](http://blog.chinaunix.net/u1/37292/showart_287681.html)

第一种常用方式:  
#模块设置，一般设置为app(生成应用程序,默认),subdirs(生成makefile文件编译subdirs指定的子文件夹),lib(生成库文件)  
TEMPLATE = app  
#指定生成的应用程序放置的目录  
DESTDIR += ../bin  
#指定生成的应用程序名  
TARGET = pksystem  
#添加或减少CONFIG的配置模块,一般采用默认即可,但是除去debug\_and\_release模块不会生成debug和release文件夹  
#CONFIG += release  
CONFIG -= debug\_and\_release  
#指定uic命令将.ui文件转化成ui\_\*.h文件的存放的目录  
UI\_DIR += forms  
#指定rcc命令将.qrc文件转换成qrc\_\*.h文件的存放目录  
RCC\_DIR += ../tmp  
#指定moc命令将含Q\_OBJECT的头文件转换成标准.h文件的存放目录  
MOC\_DIR += ../tmp  
#指定目标文件的存放目录  
OBJECTS\_DIR += ../tmp  
#程序编译时依赖的相关路径  
DEPENDPATH += . forms include qrc sources  
#头文件包含路径  
INCLUDEPATH += .  
#qmake时产生的信息,$${a}读取变量a的字符串,$$(PATH)读取环境变量PATH  
#message($$(PATH))  
#源文件编码方式  
CODECFORSRC = GBK  
# Input  
#工程中包含的头文件  
HEADERS += include/painter.h  
#工程中包含的.ui设计文件  
FORMS += forms/painter.ui  
#工程中包含的源文件  
SOURCES += sources/main.cpp sources/painter.cpp  
#工程中包含的资源文件  
RESOURCES += qrc/painter.qrc  
   
第二种常用方式,用于子文件夹的编译:  
#设定模块为子文件夹编译方式  
TEMPLATE = subdirs  
#子文件夹目录,可用\换行  
SUBDIRS = src1 \  
          src2  
  
摘自：http://blog.csdn.net/metasearch/archive/2008/03/05/2148207.aspx  
在QT中，有一个工具qmake可以生成一个makefile文件，它是由.pro文件生成而来的，.pro文件的写法如下：  
  
1. 注释  
从“#”开始，到这一行结束。  
  
2. 指定源文件  
SOURCES = \*.cpp  
  
对于多源文件，可用空格分开，如：SOURCES = 1.cpp 2.cpp3.cpp  
或者每一个文件可以被列在一个分开的行里面，通过反斜线另起一行，就像这样:  
  
SOURCES = hello.cpp  
     main.cpp  
一个更冗长的方法是单独地列出每一个文件，就像这样：  
  
    SOURCES+= hello.cpp  
    SOURCES +=main.cpp  
  
这种方法中使用“+=”比“=”更安全，因为它只是向已有的列表中添加新的文件，而不是替换整个列表。  
  
3. 指定头文件  
HEADERS = hello.h或者HEADERS += hello.h  
  
列出源文件的任何一个方法对头文件也都适用。  
  
4. 配置信息  
CONFIG用来告诉qmake关于应用程序的配置信息。  
  
    CONFIG+= qt warn\_on release  
  
在这里使用“+=”，是因为我们添加我们的配置选项到任何一个已经存在中。这样做比使用“=”那样替换已经指定的所有选项是更安全的。  
A> qt部分告诉qmake这个应用程序是使用Qt来连编的。这也就是说qmake在连接和为编译添加所需的包含路径的时候会考虑到Qt库的。  
B> warn\_on部分告诉qmake要把编译器设置为输出警告信息的。  
C> release部分告诉qmake应用程序必须被连编为一个发布的应用程序。在开发过程中，程序员也可以使用debug来替换release  
  
5. 指定目标文件名  
TARGET = filename  
  
如果不设置该项目，目标名会被自动设置为跟项目文件一样的名称  
  
6. 添加界面文件(ui)  
INTERFACES = filename.ui  
  
7. 平台相关性处理  
我们在这里需要做的是根据qmake所运行的平台来使用相应的作用域来进行处理。为Windows平台添加的依赖平台的文件的简单的作用域看起来就像这样：  
  
win32 {  
SOURCES += hello\_win.cpp  
}  
  
所以如果qmake运行在Windows上的时候，它就会把hello\_win.cpp添加到源文件列表中。如果qmake运行在其它平台上的时候，它会很简单地把这部分忽略。  
  
8. 如果一个文件不存在，停止qmake  
如果某一个文件不存在的时候，你也许不想生成一个Makefile。我们可以通过使用exists()函数来检查一个文件是否存在。我们可以通过使用error()函数把正在运行的qmake停下来。这和作用域的工作方式一样。只要很简单地用这个函数来替换作用域条件。对main.cpp文件的检查就像这样：  
  
!exists( main.cpp ) {  
  error( "No main.cpp file found")  
}  
  
“!”用来否定这个测试，比如，如果文件存在，exists( main.cpp)是真，如果文件不存在，!exists( main.cpp )是真。  
  
9. 检查多于一个的条件  
假设你使用Windows并且当你在命令行运行你的应用程序的时候你想能够看到qDebug()语句。除非你在连编你的程序的时候使用console设置，你不会看到输出。我们可以很容易地把 console添加到CONFIG行中，这样在Windows下，Makefile就会有这个设置。但是如果告诉你我们只是想在当我们的应用程序运行在 Windows下并且当debug已经在CONFIG行中的时候，添加console。这需要两个嵌套的作用域；只要生成一个作用域，然后在它里面再生成另一个。把设置放在最里面的作用域里，就像这样：  
  
win32 {  
  debug {  
     CONFIG += console  
  }  
}  
  
嵌套的作用域可以使用冒号连接起来，像这样：  
  
win32:debug {  
CONFIG += console  
}  
  
10. 摸板  
模板变量告诉qmake为这个应用程序生成哪种makefile。下面是可供使用的选择：  
  
A> app -建立一个应用程序的makefile。这是默认值，所以如果模板没有被指定，这个将被使用。  
B> lib - 建立一个库的makefile。  
C> vcapp - 建立一个应用程序的VisualStudio项目文件。  
D> vclib - 建立一个库的VisualStudio项目文件。  
E> subdirs -这是一个特殊的模板，它可以创建一个能够进入特定目录并且为一个项目文件生成makefile并且为它调用make的makefile。  
  
11. 生成Makefile  
当你已经创建好你的项目文件，生成Makefile就很容易了，你所要做的就是先到你所生成的项目文件那里然后输入：  
  
Makefile可以像这样由“.pro”文件生成：  
  
    qmake -oMakefile hello.pro  
  
对于VisualStudio的用户，qmake也可以生成“.dsp”文件，例如：  
    qmake -tvcapp -o hello.dsp hello.pro