写这篇文章的目的是为了让Qt像VC++那样，支持一个工程包含多个项目。即1个App+多个动态库。本案例以1个App+1个共享库为例：

工程源码我已上传<http://download.csdn.net/detail/libaineu2004/7188655>，欢迎下载！

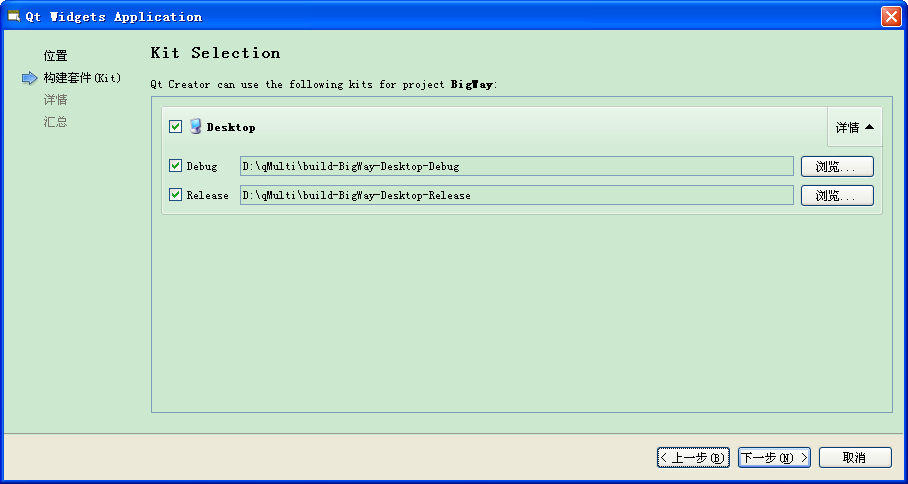
1、在D盘创建空文件夹qMulti

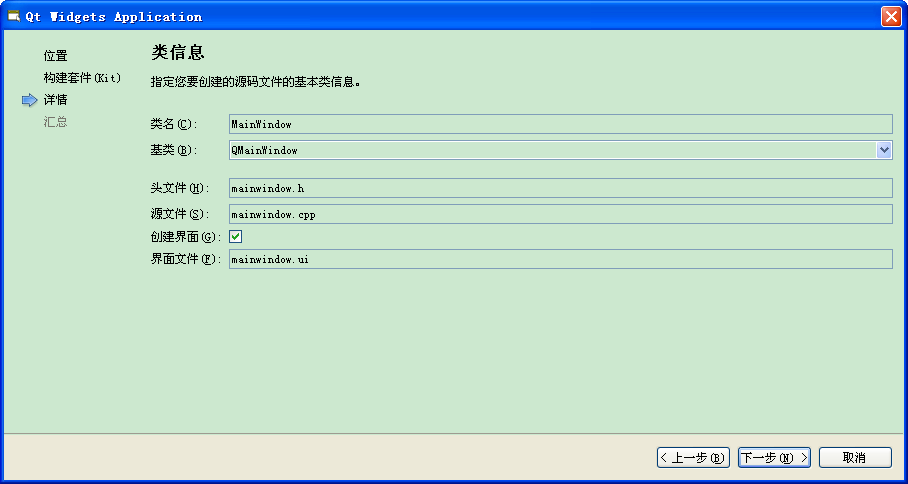
http://img.blog.csdn.net/20140413225818593

2、使用Qt Creator创建App应用程序：BigWay。注意：创建模板是App，存放路径是D:\qMulti

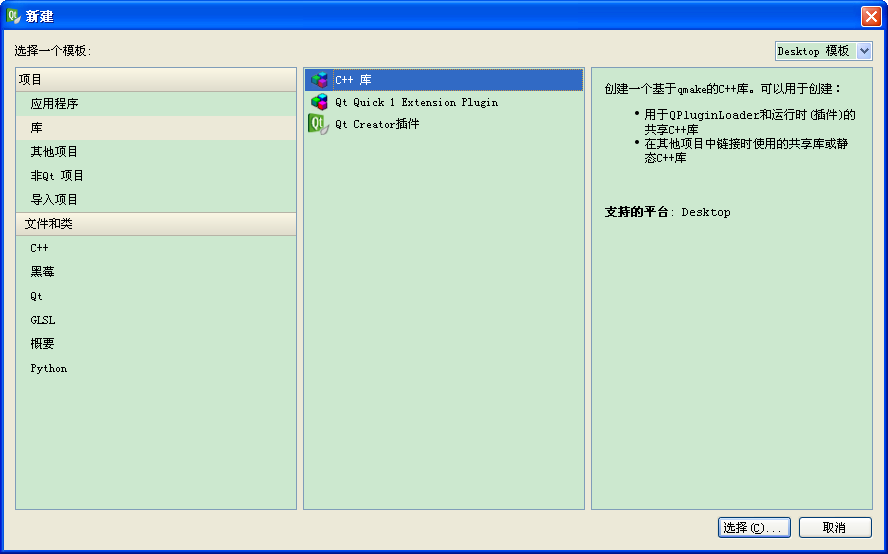
创建完成后，请关闭当前项目。

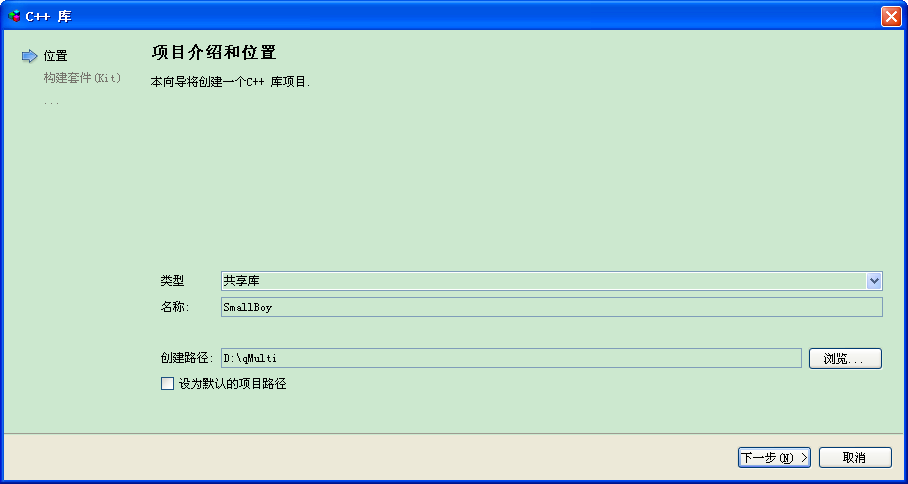


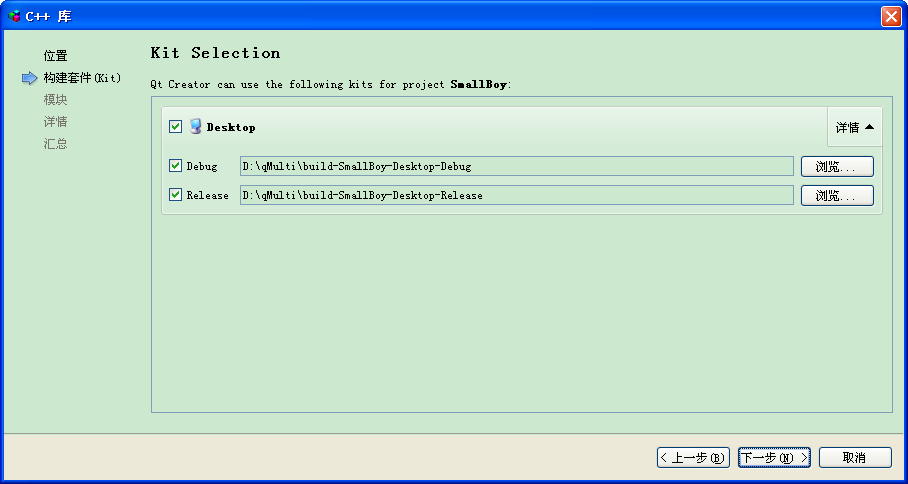
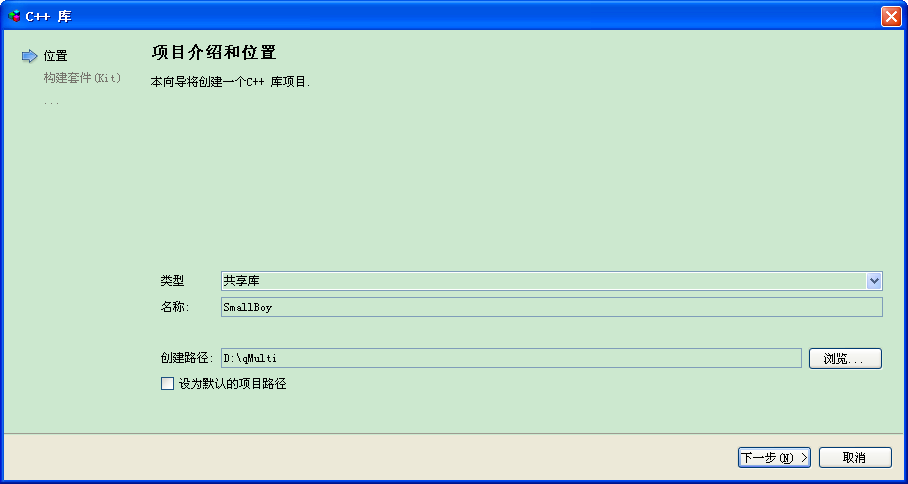


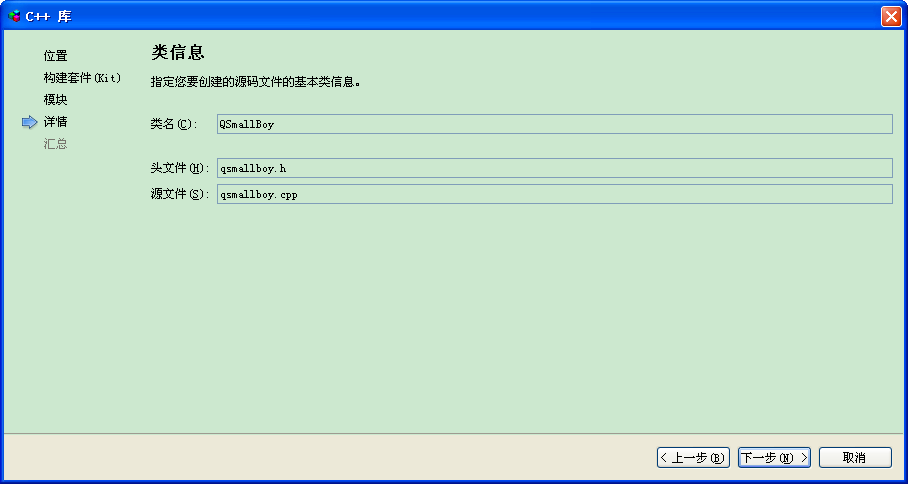


3、使用Qt Creator创建共享库项目：SmallBoy。注意：创建模板是C++ lib，存放路径是D:\qMulti



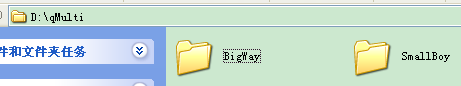






创建完成后，请关闭当前项目。

4、App和Lib创建完成之后，D:\qMulti的情况如下：



我们手动新建一个文本文件D:\qMulti\qMulti.pro，内容如下：

TEMPLATE=subdirs#表示这个是Qt跨目录,由多个子项目(如1个应用程序+多个共享库)组成的大项目

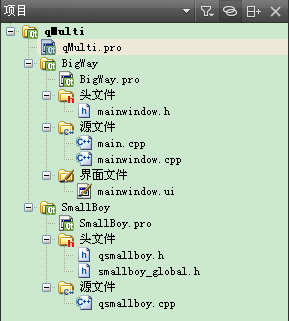
SUBDIRS=SmallBoyBigWay#大项目包含的各个子项目,其中BigWay是应用程序,SmallBoy是共享库

CONFIG+=ordered#CONFIG选项要求各个子项目按顺序编译，子目录的编译顺序在SUBDIRS中指明

到此为止，情况如下：



5、使用Qt Creator打开大项目qMulti.pro，可以看到项目树形结构如下：



（1）原始的BigWay.pro文件是这样的：

#-------------------------------------------------

#

#ProjectcreatedbyQtCreator2014-04-13T20:18:17

#

#-------------------------------------------------

QT       +=coregui

greaterThan(QT\_MAJOR\_VERSION,4):QT+=widgets

TARGET=BigWay

TEMPLATE=app

SOURCES+=main.cpp\

        mainwindow.cpp

HEADERS  +=mainwindow.h

FORMS    +=mainwindow.ui

我们将其修改为：

#-------------------------------------------------

#

#ProjectcreatedbyQtCreator2014-04-13T19:02:41

#

#-------------------------------------------------

QT+=coregui#需要的Qt库

greaterThan(QT\_MAJOR\_VERSION,4):QT+=widgets

TARGET=BigWay#表示编译后的库名字为BigWay

TEMPLATE=app#表示项目是应用程序模式

DESTDIR=../bin#定义项目编译之后生成的结果文件的存放路径

win32{#表示Windows环境的特殊设置，这个大括号不能单独放在下一行，否则编译会报错

    LIBS+=../bin/SmallBoy.lib#包含项目需要用到的共享库

}

unix{#表示Linux环境的特殊设置，这个大括号不能单独放在下一行，否则编译会报错

    LIBS+=../bin/libSmallBoy.so#包含项目需要用到的共享库，注意文件名多了关键字"lib"，扩展名是.so

}

SOURCES+=main.cpp\#项目中的所有源文件的列表

        mainwindow.cpp

HEADERS  +=mainwindow.h#项目中的所有头文件的列表

FORMS    +=mainwindow.ui#项目中的所有.ui文件（由Qt设计器生成）的列表

（2）原始的SmallBoy.pro文件是这样的：

#-------------------------------------------------

#

#ProjectcreatedbyQtCreator2014-04-13T20:24:55

#

#-------------------------------------------------

QT       -=gui

TARGET=SmallBoy

TEMPLATE=lib

DEFINES+=SMALLBOY\_LIBRARY

SOURCES+=qsmallboy.cpp

HEADERS+=qsmallboy.h\

        smallboy\_global.h

unix{

    target.path=/usr/lib

    INSTALLS+=target

}

我们将其修改为：

#-------------------------------------------------

#

#ProjectcreatedbyQtCreator2014-04-13T19:04:24

#

#-------------------------------------------------

QT-=gui#不需要Qtui库

TARGET=SmallBoy#表示编译后的库名字为SmallBoy

TEMPLATE=lib#表示项目是共享库模式

DEFINES+=SMALLBOY\_LIBRARY#项目所需的额外的预处理程序定义的列表

DESTDIR=../bin#定义项目编译之后生成的结果文件的存放路径

SOURCES+=qsmallboy.cpp#项目中的所有源文件的列表

HEADERS+=qsmallboy.h\#项目中的所有头文件的列表

        smallboy\_global.h

win32{#表示Windows环境的特殊设置，这个大括号不能单独放在下一行，否则编译会报错

}

unix{#表示Linux环境的特殊设置，这个大括号不能单独放在下一行，否则编译会报错

    target.path=/usr/lib

    INSTALLS+=target

}

6、Linux环境下使用

如果把完整的项目源码qMulti拷贝到Ubantu Linux下的Qt编译，编译过程一切OK。但是运行时会报错，说“errorwhile loading shared libraries: libSmallBoy.so.1: cannot open shared objectfile: No such file or directory”原因指的是Linux下的共享库路径配置不正确，找不到libSmallBoy.so库文件。默认的linux共享库搜索路径为/lib和/usr/lib两个目录（不包含子目录）。

解决方案有两种：

（1）把编译生成的四个.so文件全部拷贝到文件系统的路径/usr/lib之下，需要管理员权限。



再运行程序，OK。

（2）修改Linux环境变量LD\_LIBRARY\_PATH，把Qt编程生成的路径加入。

export LD\_LIBRARY\_PATH=/usr/local/lib

在终端里运行上面这行命令，再运行程序，如果正常就说明解决问题。但是该方法只是临时设置变量 LD\_LIBRARY\_PATH ，下次开机，一切设置将不复存在；

 7、换一种思路建项目

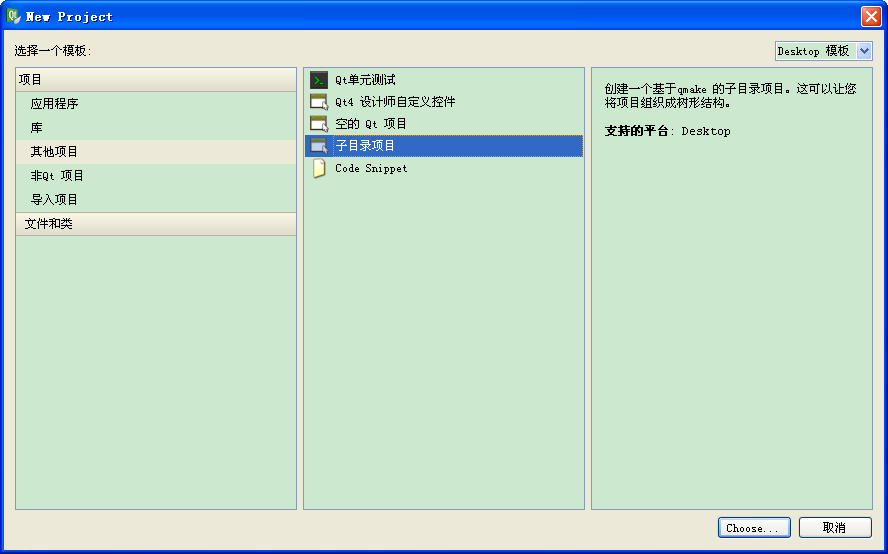
之前步骤1~4讲的是先建项目再建qMulti.pro文件。其实我们可以采取逆序，先建qMulti.pro文件，后建子项目SmallBoy和BigWay。

我们手动新建一个文本文件D:\qMulti\qMulti.pro，内容如下：

TEMPLATE=subdirs#表示这个是Qt跨目录,由多个子项目(如1个应用程序+多个共享库)组成的大项目

CONFIG+=ordered#CONFIG选项要求各个子项目按顺序编译，子目录的编译顺序在SUBDIRS中指明

或者用Qt Creator新建子目录项目qMulti：



然后用Qt Creator打开这个空的工程。右键菜单，选择“新子项目”，创建即可。(\*^\_\_^\*)

