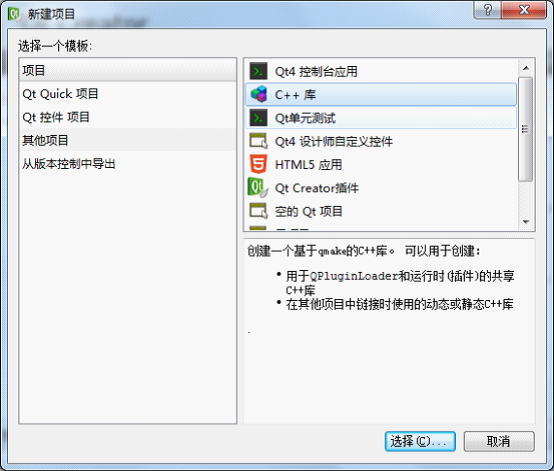
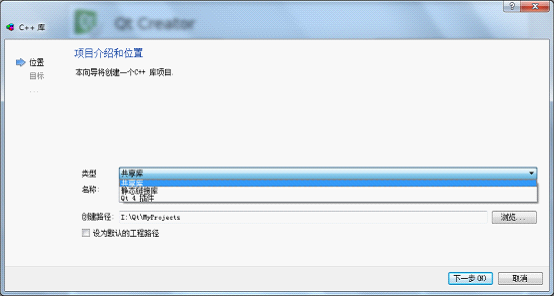
**一、创建共享库**

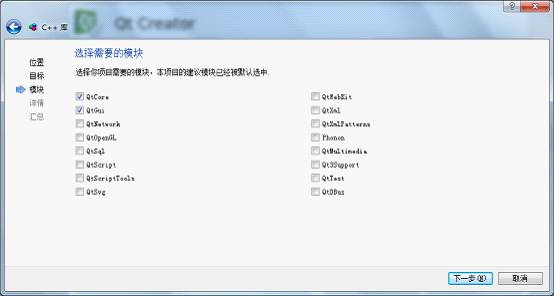
1、新其他建项目，选择C++库



2、选择共享库，并取项目名称，单击下一步。这里取名位mylib

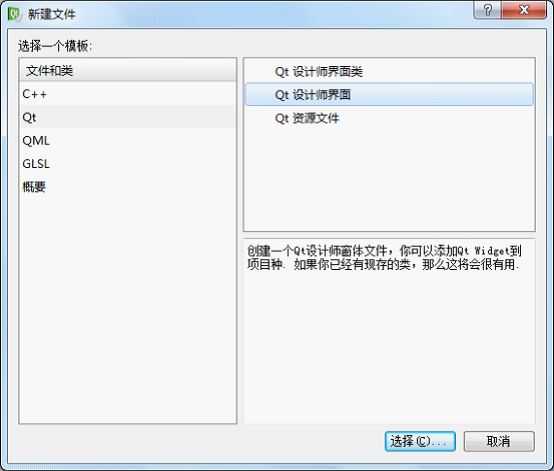


3、按默认配置单击下一步至模块选项，选择所需支持的模块。这里勾选Qtcore和QtGui。

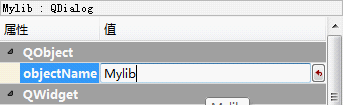


4、直接下一步至完成。现在一个共享库的工程就新建好了。

5、右击项目添加新文件，选择Qt设计师界面，以便支持可视化界面设计。



6、进入可视化界面设计，修改objectName为Mylib



7、修改mylib.h文件，代码如下：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/q496713258/article/details/6990837)

1. #ifndef MYLIB\_H
2. #define MYLIB\_H
3. #include "mylib\_global.h"
4. #include <QDialog>
5. namespace Ui
6. {
7. class Mylib;
8. }
9. class MYLIBSHARED\_EXPORT Mylib :public QDialog
10. {
11. public:
12. Mylib(QWidget \*parent = 0);
13. private:
14. Ui::Mylib \*ui;
15. };
16. #endif // MYLIB\_H

#ifndef MYLIB\_H

#define MYLIB\_H

#include "mylib\_global.h"

#include <QDialog>

namespace Ui

{

class Mylib;

}

class MYLIBSHARED\_EXPORT Mylib :public QDialog

{

public:

Mylib(QWidget \*parent = 0);

private:

Ui::Mylib \*ui;

};

#endif // MYLIB\_H

8、修改mylib.cpp文件，代码如下：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/q496713258/article/details/6990837)

1. #include "mylib.h"
2. #include "ui\_mylib.h"
3. Mylib::Mylib(QWidget \*parent):
4. QDialog(parent),
5. ui(new Ui::Mylib)
6. {
7. ui->setupUi(this);
8. }

#include "mylib.h"

#include "ui\_mylib.h"

Mylib::Mylib(QWidget \*parent):

QDialog(parent),

ui(new Ui::Mylib)

{

ui->setupUi(this);

}

9、右键单击项目，选择构建项目"mylib"。如果没有错误，构建成功后会在Debug文件夹下出现这三个文件。



mylib.dll即为后面要使用到的共享库，libmylib.a是Linux下使用的库。

至此，已生成了要使用的共享库。

**二、共享库的使用**

现在需要新建一个工程来调用我们刚才生成的共享库。这里我创建了一个名为libtest的Dialog工程。

1、修改工程文件libtest.pro，在文件尾添加如下代码：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/q496713258/article/details/6990837)

1. INCLUDEPATH += ../
2. LIBS += ../debug/lib.dll

INCLUDEPATH += ../

LIBS += ../debug/lib.dll

INCLUDEPATH 这一行就是为了让我们的 libtest 项目可以找到 mylib.h 和 mylib\_global.h 这两个文件，你需要把这里的路径替换成符合你的工程的路径。

LIBS 这一行则需要告诉编译器（注意，这里是编译器！）到哪里去找到这个 dll 文件。

这里，我的路径为：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/q496713258/article/details/6990837)

1. INCLUDEPATH += I:\Qt\MyProjects\mylib
2. LIBS += I:\Qt\MyProjects\mylib\debug\mylib.dll

INCLUDEPATH += I:\Qt\MyProjects\mylib

LIBS += I:\Qt\MyProjects\mylib\debug\mylib.dll

2、在dialog中添加一个button，单击button时调用共享库。修改dialog.cpp文件，代码如下：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/q496713258/article/details/6990837)

1. #include "dialog.h"
2. #include "ui\_dialog.h"
3. #include "mylib.h"
4. Dialog::Dialog(QWidget \*parent) :
5. QDialog(parent),
6. ui(new Ui::Dialog)
7. {
8. ui->setupUi(this);
9. }
10. Dialog::~Dialog()
11. {
12. delete ui;
13. }
14. void Dialog::on\_pushButton\_clicked()
15. {
16. Mylib w;
17. w.show();
18. w.exec();
19. }

#include "dialog.h"

#include "ui\_dialog.h"

#include "mylib.h"

Dialog::Dialog(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Dialog)

{

ui->setupUi(this);

}

Dialog::~Dialog()

{

delete ui;

}

void Dialog::on\_pushButton\_clicked()

{

Mylib w;

w.show();

w.exec();

}

3、此时运行一下，会是什么效果呢？什么都没有！提示

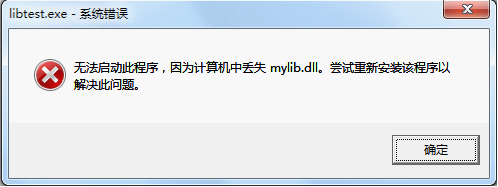
**[cpp]** [view plaincopyprint?](http://blog.csdn.net/q496713258/article/details/6990837)

1. I:\Qt\MyProjects\libtest\debug\libtest.exe 启动中...
2. I:\Qt\MyProjects\libtest\debug\libtest.exe 退出， 代码: -1073741515

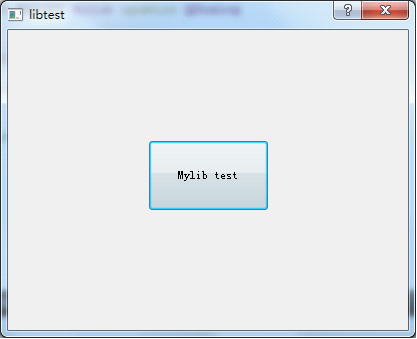
I:\Qt\MyProjects\libtest\debug\libtest.exe 启动中...

I:\Qt\MyProjects\libtest\debug\libtest.exe 退出， 代码: -1073741515

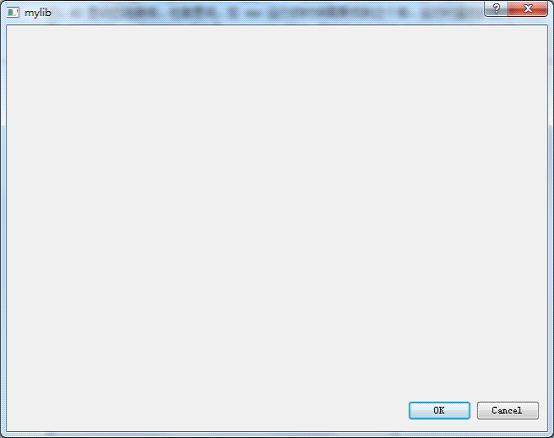
找到libtest工程目录下的libtest.exe，双击运行，弹出



没有找到mylib.dll？不是使用 LIBS 指定了吗？请注意，我们强调了，这个指定是编译期的。dll 是动态链接库，也就是说，在 exe 运行的时候需要找到这个库。运行时查找的顺序是：当前路径 -> 系统路径(通常是 system32)。所以，要把我们先前生成的这个 mylib.dll 复制到 exe 所在目录，然后直接双击一下这个 exe 文件。程序运行如下



单击Mylib test按钮，共享库mylib的dialog也弹出来啦~~~



到这里，共享库的使用也成功了。

**三、更新dll**

也许有人要问了，共享库的创建和使用都这么麻烦，用它有什么好处呢。

共享库的一个好处是可以动态加载，也就是说，如果你需要升级程序，那么就要简单的替换掉这个 dll 就好了，不需要要求用户重新安装全部文件。当然，这些 dll 也是有缺点的：动态加载的东西肯定会比静态编译的东西效率低一些。不过在现在的硬件环境下，这点性能损失已经可以忽略不计了。

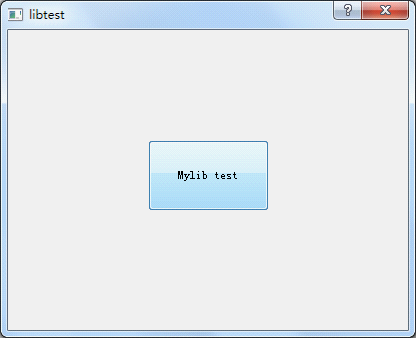
好，那么我们现在来体验一下如何更新dll来达到升级程序的目的。

1、修改mylib.cpp，在共享库的dialog中添加一个button。

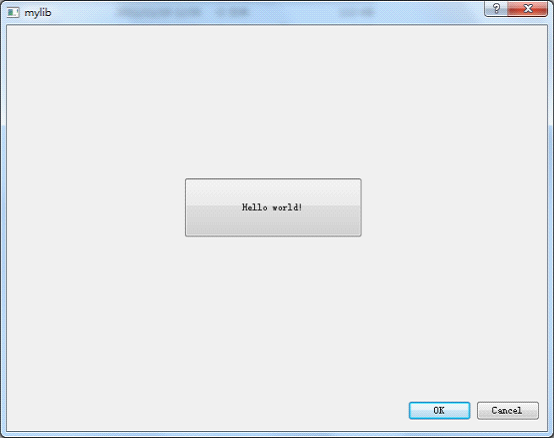
2、右键单击项目mylib，选择重新构建项目"mylib"

3、将libtest项目目录下的mylib.dll替换新的mylib.dll。

4、运行libtest.exe



这时单击Mylib test按钮，效果如何呢？



至此，我们在没有修改程序的情况下，通过替换dll达到了改变程序运行效果的目的。也就是说我们可以通过更新dll文件来完成对程序的升级了