Remote Object Compiler

The remote object compiler takes a QObject subclass (known as candidate class) in the form of a path, header and implementation names, with some additional arguments. During run time, it first backs up your original project and implementation. It then builds your candidate class into an object, it runs QMetaObject's introspection and generates a new version of the implementation which is named using the convention of rpc_<original_name>.cpp.

The new implementation name is substituted in the project file you specified as a program argument. When rebuilt, your client application is ready to make RPC calls to remote instances of the same class. Once this step is complete, you can bind an instance of the modified candidate object to a remote instance of the real, unmodified object. When you bind on the client, you specify the name of the remote object, as per QObject's objectName() and setObjectName() and the IP/host name on which the remote object exists, with the port that the remote object has been set to listen on.

ROC is designed to run from the project directory for your client application, although you can theoretically run it from anywhere, providing it is given QTDIR and the location of make as arguments. It automatically creates a subdirectory called "subproject" in which it copies some source files and generates a program called "loader". When loader is invoked by roc, it instantiates your candidate object for introspection before automatically generating rpc_<yourclass>.cpp and modifying yourproject.pro in the client project directory.

For an example of this, see tests/rpctest for a working test harness employing a roc'ed class. To run roc on your own project, use the process outlined below.

```
roc -prefix <path to source>

        -pro <project file name>

        -si <implementation name>.cpp

        -sh <header name>.h

        -class <source class name>

        -qtdir [qt-directory]

        -make [path-to-make]

        [-inc [-I[include-path] -i[include-name]]...]

        [-lib [-L[library-path] -l[library-name]]...]

        [-dep [-dpath[dependency-path] -d[dependency-name]]...]
```

As an example, I used the following arguments when invoking roc on my test project

```
roc -prefix /home/matthew/projects/qt-rpc-framework.master/tests/testrpc -pro testrpc.pro -si rpctestclass.cpp -sh rpctestclass.h -class RpcTestClass -qtdir /home/matthew/dev/qt5/install -make /usr/bin/make
```

The target project in /home/matthew/projects/qt-rpc-framework.master/tests/testrpc is the testrpc autotest. The rest of the arguments should be self explanatory, however, you should include any dependencies, libraries or headers that your candidate QObject subclass requires in order to build successfully. These will be added to the loader app that is built and run internally by roc.