作业8 Qt MQTT开发库移植与测试

布置时间：5月26日

收集时间：6月1日24：00

提交方式：姓名\_学号\_作业8，如：平毅\_2009060707\_作业8.doc

内容：

1. 在Deepin操作系统（可在Windows版本上先进行开发），利用GitHub Desktop下载Qt MQTT官方库文件包，根据Qt安装版本（课堂中演示的版本位5.12.9）选择对应的MQTT库文件包版本两者需对应。

2. 打开文件包中的qtmqtt.pro工程文件，分别执行build和run qmake（汉化版本名称分别位编译和运行），生成编译输出文件夹，本例中位：build-qtmqtt-Desktop\_Qt\_5\_12\_9\_MinGW\_64\_bit-Debug。

3. 分别将该文件夹中的lib文件夹内6个文件，bin文件夹内2个文件，mkspecs/modules内1个文件复制至C:\Qt\Qt5.12.9\5.12.9\mingw73\_64下对应文件夹（注意实际操作可能存在不同安装路径）。把示例代码（C:\Users\pyhom\Documents\GitHub\qtmqtt\examples）mqtt整个目录copy至QT安装目录（C:\Qt\Qt5.12.9\Examples\Qt-5.12.9）下。

**针对5.12.9的前期设置已完成，紧接着完成代码测试，新建MQTT\_Test工程。**

1.需要在pro文件中添加 QT +=mqtt(注意emqx/qmqtt版本相同，该版本需添加QT +=qtmqtt)

2 进行一次编译无报错后继续添加库文件。

3. 依据5.12.9示例程序，需在mainWindow.h中添加#include <QMqttClient> (课堂上也举例了emqx/qmqtt版本示例，为添加#include “mqtt/qmqtt.h”,#include "mqtt/qmqtt\_client.h"，编译过程中需调整库文件符号)

4. 在private中声明mqttclient对象，5.12.9的构造方法为：QMqttClient \*myMqttclient;（emqx/qmqtt版本，为 QMQTT::Client \*myMqttclient;）

5. 完成编译，如无错则编译成功，可进行MQTT工程的二次开发。

6. 可直接运行GitHub库文件夹中simpleclient示例，完成MQTT功能测试。



图1 SimpleClient示例运行界面图

答：mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QtCore/QDateTime>

#include <Qtmqtt/QMqttClient.h>

#include <QtWidgets/QMessageBox>

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

m\_client = new QMqttClient(this);

m\_client->setHostname(ui->lineEditHost->text());

m\_client->setPort(ui->spinBoxPort->value());

connect(m\_client, &QMqttClient::stateChanged, this, &MainWindow::updateLogStateChange);

connect(m\_client, &QMqttClient::disconnected, this, &MainWindow::brokerDisconnected);

connect(m\_client, &QMqttClient::messageReceived, this, [this](const QByteArray &message, const QMqttTopicName &topic) {

const QString content = QDateTime::currentDateTime().toString()

+ QLatin1String(" Received Topic: ")

+ topic.name()

+ QLatin1String(" Message: ")

+ message

+ QLatin1Char('\n');

ui->editLog->insertPlainText(content);

});

connect(m\_client, &QMqttClient::pingResponseReceived, this, [this]() {

ui->buttonPing->setEnabled(true);

const QString content = QDateTime::currentDateTime().toString()

+ QLatin1String(" PingResponse")

+ QLatin1Char('\n');

ui->editLog->insertPlainText(content);

});

connect(ui->lineEditHost, &QLineEdit::textChanged, m\_client, &QMqttClient::setHostname);

connect(ui->spinBoxPort, QOverload<int>::of(&QSpinBox::valueChanged), this, &MainWindow::setClientPort);

updateLogStateChange();

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**on\_buttonConnect\_clicked**()

{

if (m\_client->state() == QMqttClient::Disconnected) {

ui->lineEditHost->setEnabled(false);

ui->spinBoxPort->setEnabled(false);

ui->buttonConnect->setText(tr("Disconnect"));

m\_client->connectToHost();

} else {

ui->lineEditHost->setEnabled(true);

ui->spinBoxPort->setEnabled(true);

ui->buttonConnect->setText(tr("Connect"));

m\_client->disconnectFromHost();

}

}

void MainWindow::**on\_buttonQuit\_clicked**()

{

QApplication::quit();

}

void MainWindow::**updateLogStateChange**()

{

const QString content = QDateTime::currentDateTime().toString()

+ QLatin1String(": State Change")

+ QString::number(m\_client->state())

+ QLatin1Char('\n');

ui->editLog->insertPlainText(content);

}

void MainWindow::**brokerDisconnected**()

{

ui->lineEditHost->setEnabled(true);

ui->spinBoxPort->setEnabled(true);

ui->buttonConnect->setText(tr("Connect"));

}

void MainWindow::**setClientPort**(int p)

{

m\_client->setPort(p);

}

void MainWindow::**on\_buttonPublish\_clicked**()

{

if (m\_client->publish(ui->lineEditTopic->text(), ui->lineEditMessage->text().toUtf8()) == -1)

QMessageBox::critical(this, QLatin1String("Error"), QLatin1String("Could not publish message"));

}

void MainWindow::**on\_buttonSubscribe\_clicked**()

{

auto subscription = m\_client->subscribe(ui->lineEditTopic->text());

if (!subscription) {

QMessageBox::critical(this, QLatin1String("Error"), QLatin1String("Could not subscribe. Is there a valid connection?"));

return;

}

}

void MainWindow::**on\_buttonPing\_clicked**()

{

ui->buttonPing->setEnabled(false);

m\_client->requestPing();

}

答：mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <Qtmqtt/QMqttClient.h>

QT\_BEGIN\_NAMESPACE

namespace **Ui** {

class **MainWindow**;

}

QT\_END\_NAMESPACE

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

explicit **MainWindow**(QWidget \*parent = 0);

~***MainWindow***();

public slots:

void **setClientPort**(int p);

private slots:

void **on\_buttonConnect\_clicked**();

void **on\_buttonQuit\_clicked**();

void **updateLogStateChange**();

void **brokerDisconnected**();

void **on\_buttonPublish\_clicked**();

void **on\_buttonSubscribe\_clicked**();

void **on\_buttonPing\_clicked**();

private:

Ui::MainWindow \*ui;

QMqttClient \*m\_client;

};

#endif // MAINWINDOW\_H

