

昵称：菜头大大
园龄：2年3个月
粉丝：32
关注：9
+加关注

< 2019年11月 >						
日	一	二	三	四	五	六
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

c语言中的文件格式化读写函数
fscan和fprintf函数(1)
c语言中的转义序列(1)
linux进程(1)
linux内核链表(1)
linux网络编程(1)
linux无名管道(1)
linux下文件和目录(1)
linux有名管道(1)
linux中文件I/O操作（系统I/O）(1)
unix下的文件和目录详解以及操作方法(1)
更多

积分与排名

积分 - 49760
排名 - 13702

随笔分类

apache(1)
c++(1)
c语言学习(3)
linux数据库(4)
linux小技巧(1)
linux学习(6)
QT(15)
stm32(5)
unix环境高级编程学习(4)
zigbee(1)

面试题分析(5)
小问题(1)
疑难杂症(2)

随笔档案

2018年7月(1)
2018年4月(1)
2018年3月(9)

博客园 首页 新随笔 联系 订阅 管理

QT之TCP通信

QT中可以通过TCP协议让服务器和客户端之间行通信。所以下面我就围绕服务器和客户端来写。
这是我们写服务器和客户端的具体流程：

A、服务器：

- 1.创建QTcpServer对象
- 2.启动服务器（监听）调用成员方法listen（QHostAddress::Any,端口号）
- 3.当有客户端链接时候会发送newConnection信号，触发槽函数接受链接（得到一个与客户端通信的套接字QTcpSocket）
- 4.QTcpsocket发送数据用成员方法write，
- 5.读数据当客户端有数据来，QTcpSocket对象就会发送readyRead信号，关联槽函数读取数据

B、客户端：

- 1.创建QTcpSocket对象
- 2.链接服务器connectToHost(QHostAddress("ip"),端口号)
- 3.QTcpsocket发送数据用成员方法write，
- 4.读数据当对方有数据来，QTcpSocket对象就会发送readyRead信号，关联槽函数读取数据

我们需要调用到的头文件有两个：

```
#include <QTcpServer>  
#include <QTcpSocket>
```

我们先要在工程文件中加入network

```
QT += core gui network
```

下面我们来看看服务器程序步骤：

1、初始化服务器server对象

```
mServer = new QTcpServer();
```

2、启动监听服务器

```
mServer->listen(QHostAddress::Any,9988);//9988为端口号
```

3、当有客户端链接时候会发送newConnection信号，触发槽函数接受链接（得到一个与客户端通信的套接字QTcpSocket）

```
connect(mServer,SIGNAL(newConnection()),this,SLOT(new_client()));  
mSocket = mServer->nextPendingConnection();//与客户端通信的套接字
```

4、发送数据

```
mSocket->write(msg.toUtf8());
```

5、读数据当客户端有数据来，QTcpSocket对象就会发送readyRead信号，关联槽函数读取数据

```
connect(mSocket,SIGNAL(readyRead()),this,SLOT(read_client_data()));
```

6、连接多个客户端

2018年2月(5)
2018年1月(9)
2017年12月(3)
2017年11月(7)
2017年10月(1)
2017年9月(7)
2017年8月(5)
2017年7月(1)

文章分类

QT

最新评论

1. Re:QT串口通信
我想问一下，为什么我下位机接收到的字节很多都变为了3F
--天天不想说
2. Re:qt中建立图片资源文件
好文章，说的很清楚，感谢分享
--jsxyhelu
3. Re:qt中建立图片资源文件
谢谢，明白了怎么添加图片了
--白白大黄小狐狸一只耳
4. Re:QT制作一个图片播放器
这个ui是把菜单栏和工具栏啥的都去掉了吗？
--//探索者//
5. Re:QT之UDP通信
溜溜溜
--DreamDog

阅读排行榜

1. QT串口通信(27201)
2. c语言中的文件格式化读写函数fscanf和printf函数(18776)
3. QT之UDP通信(16318)
4. QT之TCP通信(15041)
5. 使用qt制作一个简单的计算器(10905)

评论排行榜

1. QT制作一个图片播放器(6)
2. linux下修改rm命令防止误删除(2)
3. qt中建立图片资源文件(2)
4. 面试题第一弹(2)
5. QT之UDP通信(1)

推荐排行榜

1. QT串口通信(7)
2. QT之TCP通信(5)
3. qt中建立图片资源文件(2)
4. QT之UDP通信(1)
5. 初识QT(1)

```
//可以实现同时读取多个客户端发送过来的消息
QTcpSocket *obj = (QTcpSocket*)sender();
```

7、检测掉线

```
connect(mSocket, SIGNAL(disconnected()), this, SLOT(client_dis())); //检测掉线信号
```

下面是服务器的实现的具体代码：



```
1 #include "tcpserver.h"
2 #include "ui_tcpserver.h"
3 #include <QDebug>
4 TcpServer::TcpServer(QWidget *parent) :
5     QMainWindow(parent),
6     ui(new Ui::TcpServer)
7 {
8     ui->setupUi(this);
9     //初始化服务器server对象
10    mServer = new QTcpServer();
11    //关联客户端连接信号newConnection
12    connect(mServer, SIGNAL(newConnection()), this, SLOT(new_client())); //连接客户端
13    //启动服务器监听
14    mServer->listen(QHostAddress::Any, 9988);
15
16 }
17
18 TcpServer::~TcpServer()
19 {
20     delete ui;
21 }
22
23 void TcpServer::new_client()
24 {
25     qDebug() << "新客户段连接";
26     mSocket = mServer->nextPendingConnection(); //与客户端通信的套接字
27     //关联接收客户端数据信号readyRead信号 (客户端有数据就会发readyRead信号)
28     connect(mSocket, SIGNAL(readyRead()), this, SLOT(read_client_data()));
29     //检测掉线信号
30     connect(mSocket, SIGNAL(disconnected()), this, SLOT(client_dis()));
31
32 }
33
34 void TcpServer::read_client_data()
35 {
36     //可以实现同时读取多个客户端发送过来的消息
37     QTcpSocket *obj = (QTcpSocket*)sender();
38     QString msg = obj->readAll();
39     qDebug() << msg;
40 }
41
42 void TcpServer::client_dis()
43 {
44     QTcpSocket *obj = (QTcpSocket*)sender(); //掉线对象
45     qDebug() << obj->peerAddress().toString(); //打印出掉线对象的ip
46 }
```



说完服务器那我们继续来看看客户端是怎么实现的：

1、创建QTcpSocket对象

```
mSocket = new QTcpSocket();
```

2、链接服务器connectToHost(QHostAddress("ip"),端口号), 连接服务器ip和端口号

```
mSocket->connectToHost(ui->ipEdit->text(), ui->portEdit->text().toInt()); //ui->ipEdit->text():  
ip, ui->portEdit->text().toInt(): 端口号
```

3、发送数据

```
//取发送信息编辑框内容  
QString msg = ui->sendEdit->toPlainText();  
mSocket->write(msg.toUtf8()); //转编码
```

4、检测链接成功信号关联槽函数

```
connect(mSocket, SIGNAL(connected()), this, SLOT(connect_suc()));
```


5、检测掉线信号

```
connect(mSocket, SIGNAL(disconnected()), this, SLOT(client_dis()));
```

6、服务器和客户端关闭都可以使用close

```
mSocket->close();
```

这是客户端实现的具体代码

```

1 #include "tcpclient.h"
2 #include "ui_tcpclient.h"
3 #include <QDebug>
4 TcpClient::TcpClient(QWidget *parent) :
5     QMainWindow(parent),
6     ui(new Ui::TcpClient)
7 {
8     ui->setupUi(this);
9     //初始化套接字对象
10    mSocket = new QTcpSocket();
11    //关联数据信号
12    connect(mSocket, SIGNAL(readyRead()), this, SLOT(read_data()));
13
14 }
15
16 TcpClient::~TcpClient()
17 {
18     delete ui;
19 }
20
21 void TcpClient::read_data()
22 {
23     QString msg = mSocket->readAll();
24     qDebug() << msg;
25 }
26
27 void TcpClient::on_btn_connectServer_clicked()
28 {
29     //检测链接成功信号关联槽函数
30    connect(mSocket, SIGNAL(connected()), this, SLOT(connect_suc()));
31    //检测掉线信号
32    connect(mSocket, SIGNAL(disconnected()), this, SLOT(client_dis()));
33    //连接服务器, 设置ip和端口号
34    mSocket->connectToHost(ui->ipEdit->text(), ui->portEdit->text().toInt());
```

```

35
36 }
37
38 void TcpClient::on_btn_send_clicked()
39 {
40     //取发送信息编辑框内容
41     QString msg = ui->sendEdit->toPlainText();
42     mSocket->write(msg.toUtf8()); //转编码
43 }
44
45 void TcpClient::connect_suc()
46 {
47     ui->btn_connectServer->setEnabled(false); //如果连接成功则连接按钮不能按下
48 }
49 void TcpClient::client_dis()
50 {
51     ui->btn_connectServer->setEnabled(true); //如果连接没有成功则连接按钮还可以按下
52 }

```

这是服务器和客户端分开两个文件夹写的程序，在这里我也实现了服务器和客户端写在同一个文件中

具体代码如下：

头文件：tcpapp.h

```

1 #ifndef TCPAPP_H
2 #define TCPAPP_H
3
4 #include <QMainWindow>
5 #include <QTcpServer>
6 #include <QTcpSocket>
7 #include <QHostAddress>
8 #include <QFile>
9 #include <QTimer>
10 #include <QMessageBox>
11 namespace Ui {
12 class TcpApp;
13 }
14
15 class TcpApp : public QMainWindow
16 {
17     Q_OBJECT
18
19 public:
20     explicit TcpApp(QWidget *parent = 0);
21     ~TcpApp();
22
23 private slots:
24     void on_severRB_clicked(); //选择作为服务器
25
26     void on_clientRB_clicked(); //选择作为客户端
27
28     void on_StartBt_clicked(); //启动服务器或链接客户端
29
30     void on_closeBt_clicked(); //关闭服务器或断开客户端
31

```

```

32 void on_onlineUserList_doubleClicked(const QModelIndex &index); //选择给哪个客户端发送数据
33
34 void on_autoCB_clicked(bool checked); //选择自动发送还是手动发送
35
36 void on_sendMsgBt_clicked(); //发送信息
37
38 //服务器
39 void accept_connect(); //与newconnection信号关联
40 void recv_data(); //接收数据
41
42 void auto_time_send(); //定时器定时发送数据
43
44 void client_disconnect(); //关联掉线信号
45 void connect_suc(); //检测客户端连接成功信号
46
47 void on_clearRcvBt_clicked();
48
49 void on_clearSendBt_clicked();
50
51 private:
52 Ui::TcpApp *ui;
53 QTimer *mTimer; //定时发送数据
54 QTcpServer *mServer;
55 QTcpSocket *mSocket;
56 QVector<QTcpSocket*> clients; //存储所有在线客户端 (容器)
57
58 bool isServer; //标志位, true为服务器, false为客户端
59
60 //保存接收和发送数据的字节数
61 quint64 recvSize;
62 quint64 sendSize;
63
64 qint16 onNum;
65 bool isCheckServer; //判断是否选择了服务器
66 bool isCheckClient; //判断是否选择了客户端
67
68
69 };
70
71 #endif // TCPAPP_H

```

源文件: tcpapp.cpp

```

1 #include "tcpapp.h"
2 #include "ui_tcpapp.h"
3
4 TcpApp::TcpApp(QWidget *parent) :
5     QMainWindow(parent),
6     ui(new Ui::TcpApp),
7     onNum(0)
8 {
9     ui->setupUi(this);
10     recvSize = 0;
11     sendSize = 0;
12     //初始化定时器
13     mTimer = new QTimer();
14     connect(mTimer, SIGNAL(timeout()), this, SLOT(auto_time_send()));

```

```
15 }
16
17 TcpApp::~TcpApp()
18 {
19     delete ui;
20 }
21
22 //与newconnection信号关联
23 void TcpApp::accept_connect()
24 {
25     mSocket = mServer->nextPendingConnection(); //返回与客户端连接通信的套接字
26
27     //关联接收数据信号
28     connect(mSocket, SIGNAL(readyRead()), this, SLOT(recv_data()));
29     //关联掉线信号
30     connect(mSocket, SIGNAL(disconnected()), this, SLOT(client_disconnect()));
31
32     //上线用户添加到客户列表容器
33     clients.append(mSocket);
34     //把用户添加到界面列表中
35     QString ip = mSocket->peerAddress().toString().remove("::ffff:"); //去除客户端中多余的字符
36     ui->onlineUserList->addItem(ip);
37
38     //在线数量添加
39     onNum++;
40     ui->onlineUserCount->setText(QString::number(onNum)); //显示在线数
41
42 }
43
44 //接收数据
45 void TcpApp::recv_data()
46 {
47     QTcpSocket *obj = (QTcpSocket*)sender();
48     //获取发送数据端的IP
49     QString ip = obj->peerAddress().toString();
50     ip.remove("::ffff:");
51     QString msg = obj->readAll();
52     ui->receiveList->addItem(ip+"."+msg); //显示接收到的数据
53     recvSize += msg.size(); //统计接收到的数据的字节数
54     ui->receiveNumLabel->setText(QString::number(recvSize));
55 }
56
57 void TcpApp::client_disconnect()
58 {
59     QTcpSocket *obj = (QTcpSocket*)sender(); //获取掉线对象
60     if(isServer)
61     {
62         int row = clients.indexOf(obj); //找到掉线对象的内容所在的行
63         QListWidgetItem *item = ui->onlineUserList->takeItem(row); //从界面列表中去除找到的一行内容
64         delete item;
65         clients.remove(row); //从容器中删除对象
66
67         //掉线时删除在线数量
68         onNum--;
69         ui->onlineUserCount->setText(QString::number(onNum));
70     }
71     else
72     {
73         ui->startBt->setEnabled(true); //断开连接的时候重新启用开始按钮
```

```
74     }
75 }
76
77
78 //客户端连接成功
79 void TcpApp::connect_suc()
80 {
81     ui->StartBt->setEnabled(false); //连接成功则禁用开始按钮
82 }
83 //定时器定时发送数据
84 void TcpApp::auto_time_send()
85 {
86     quint64 len = mSocket->write(ui->sendMsgEdit->toPlainText().toUtf8());
87     if(len > 0)
88     {
89         sendSize += len; //统计发送的字节数
90         ui->sendNumLabel->setText(QString::number(sendSize)); //把发送的字节数显示到sendNumLabel上
91     }
92 }
93 }
94
95 //选择作为服务器
96 void TcpApp::on_severRB_clicked()
97 {
98     this->isCheckServer = true;
99     this->isServer = true;
100    //获取本地ip显示在IpEdit中
101    ui->IpEdit->setText(QHostAddress(QHostAddress::LocalHost).toString());
102    ui->IpEdit->setEnabled(false); //关闭ip输入编辑器
103    this->isCheckClient = false;
104
105 }
106
107 //选择作为客户端
108 void TcpApp::on_clientRB_clicked()
109 {
110     this->isCheckClient = true;
111     this->isServer = false;
112     ui->IpEdit->setEnabled(true); //打开ip输入编辑器
113     this->isCheckServer = false;
114
115 }
116
117 //启动服务器或者链接服务器
118 void TcpApp::on_StartBt_clicked()
119 {
120     if(isServer) //服务器
121     {
122         mServer = new QTcpServer();
123         //关联新客户端链接信号
124         connect(mServer, SIGNAL(newConnection()), this, SLOT(accept_connect()));
125         mServer->listen(QHostAddress::Any, ui->PortEdit->text().toInt()); //启动服务器监听
126         ui->StartBt->setEnabled(false); //开始按钮禁用
127     }
128     if(isServer == false) //客户端
129     {
130         mSocket = new QTcpSocket();
131         //检测链接成功信号
132         connect(mSocket, SIGNAL(connected()), this, SLOT(connect_suc()));
```

```

133         //设置服务器的 ip和端口号
134         mSocket->connectToHost(ui->IpEdit->text(),ui->PortEdit->text().toInt());
135
136
137         //关联接收数据信号
138         connect(mSocket,SIGNAL(readyRead()),this,SLOT(recv_data()));
139         //关联掉线信号
140         connect(mSocket,SIGNAL(disconnected()),this,SLOT(client_disconnect()));
141     }
142
143     if(isCheckServer == false && isCheckClient == false)//如果两个都没选择
144     {
145         QMessageBox::warning(this,"提示","请选择服务器或者客户端");
146         ui->StartBt->setEnabled(true);
147         return;
148     }
149
150     if(isCheckServer)//选择了服务器
151     {
152         if(ui->PortEdit->text().isEmpty() || ui->PortEdit->text() == "请输入端口号")
153         {
154             QMessageBox::warning(this,"提示","请输入端口号");
155             ui->StartBt->setEnabled(true);
156             return;
157         }
158     }
159
160     if(isCheckClient)//选择了客户端
161     {
162         if(ui->IpEdit->text().isEmpty() || ui->IpEdit->text() == "请输入ip" || ui->IpEdit->
>text() == "请输入端口号")
163         {
164             QMessageBox::warning(this,"提示","请输入ip和端口号");
165             ui->StartBt->setEnabled(true);
166             return;
167         }
168     }
169
170 }
171
172 //关闭服务器或者断开
173 void TcpApp::on_closeBt_clicked()
174 {
175     if(isServer)//服务器
176     {
177         for(int i=0;i<clients.count();i++)
178         {
179             clients.at(i)->close();//关闭所有客户端
180         }
181
182         //关闭所有服务器之后开始按钮才能启用
183         mServer->close();
184         ui->StartBt->setEnabled(true);
185     }
186     else //客户端
187     {
188         mSocket->close();//关闭客户端
189         ui->StartBt->setEnabled(true);//启用开始按钮
190     }
191

```



```
192 }
193
194 //双击选择要发送的客户端
195 void TcpApp::on_onlineUserList_doubleClicked(const QModelIndex &index)
196 {
197     mSocket = clients.at(index.row());
198
199 }
200
201 //自动发送数据
202 void TcpApp::on_autoCB_clicked(bool checked)
203 {
204     if(checked)
205     {
206         {
207             if(ui->autoTimeEdit->text().toInt() <= 0)
208             {
209                 QMessageBox::warning(this, "提示", "请输入时间值ms");
210                 ui->autoCB->setChecked(false); //把按钮重新置于没选中的状态
211                 return;
212             }
213             mTimer->start(ui->autoTimeEdit->text().toInt()); //启动定时器
214         }
215     else
216     {
217         mTimer->stop(); //停止定时器
218     }
219
220 }
221
222 //手动发送数据
223 void TcpApp::on_sendMsgBt_clicked()
224 {
225     auto_time_send();
226
227 }
228
229 //清空接收区
230 void TcpApp::on_clearRcvBt_clicked()
231 {
232     ui->receiveNumLabel->clear();
233     this->recvSize = 0;
234     ui->receiveNumLabel->setText(QString::number(recvSize));
235 }
236
237 //清空发送区
238 void TcpApp::on_clearSendBt_clicked()
239 {
240     ui->sendNumLabel->clear();
241     this->sendSize = 0;
242     ui->sendNumLabel->setText(QString::number(sendSize));
243 }
```



界面文件tcpapp.ui如下图



此外这里还使用到了容器，在这里讲讲容器的使用

1、定义容器对象

```
QVector<QTcpSocket*> clients; //存储所有在线客户端（容器）
解释: QTcpSocket* 容器的类型
clients 容器名
```

2、往容器中添加成员

```
//上线用户添加到客户列表容器
clients.append(mSocket);
```

3、寻找某个成员在容器中位置

```
int row = clients.indexOf(obj); //找到掉线对象的内容所在的行
```

4、从容器中删除成员

```
clients.remove(row); //从容器中删除成员
```

鉴于本人才疏学浅，所以其中不免有遗漏或者错误，恳请各位博友批评指正。

分类: [QT](#)

好文要顶 关注我 收藏该文





菜头大大
关注 - 9
粉丝 - 32
[+加关注](#)

5 0

« 上一篇: [qt多线程的使用方法](#)
» 下一篇: [QT之UDP通信](#)

posted @ 2017-12-10 23:47 菜头大大 阅读(15042) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

- 注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，访问 [网站首页](#)。**
- 【推荐】腾讯云海外1核2G云服务器低至2折，半价续费券限量免费领取！
 - 【活动】京东云服务器_云主机低于1折，低价高性能产品备战双11
 - 【推荐】超50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库
 - 【培训】马士兵老师强势回归！Java线下课程全免费，双十一大促！
 - 【推荐】天翼云双十一翼降到底，云主机11.11元起，抽奖送大礼

【福利】个推四大热门移动开发SDK全部免费用一年，限时抢！
【推荐】流程自动化专家UiBot，全套体系化教程成就高薪RPA工程师

京东云

亚马逊“可扩展平台”云迁移“云”引擎

11.11 京东云 年终采购季

云主机
最低 1 折

每人限购三台

立即秒杀

- 相关博文:
- QT之UDP通信
 - QT串口通信
 - 基于Tcp协议的简单Socket通信实例（JAVA）
 - 【Socket编程】通过Socket实现TCP编程
 - 界面编程之QT的Socket通信20180730
- » 更多推荐...

腾讯云

11.11 智慧上云

爆品限时购
云服务器 1核2G 首年 88元

- 最新 IT 新闻:
- 中国女排将迎史上最大一波赞助：不光有联想集团
 - 一代经典回归 摩托罗拉Razr 2019国行版亮相
 - Oculus CTO、传奇程序员John Carmack宣布离职：我要去搞AI了！
 - 16999元的华为折叠屏Mate X开卖了！秒没！
 - 假设有一把刀，原子级别锋利，从人体砍过，人会不会死？
- » 更多新闻...