Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(ФГБОУ ВО «СибГУТИ»)

*Кафедра прикладной математики и кибернетики*

Курсовая работа по дисциплине

«Операционные системы»

Выполнил: студент гр. ИП-513 Санин И. В.

Проверил:

ассистент

Бочкарев Б. В.

Новосибирск

2017

Оглавление

[1. Постановка задачи 4](#_Toc501283089)

[2. Реализация 4](#_Toc501283090)

[2.1. «Клиент» 4](#_Toc501283091)

[2.2. «Сервер» 6](#_Toc501283092)

# Постановка задачи

Разработка многопользовательского «чата» на основе сокетов.

# Реализация

Для реализации данной задачи использовалась платформа Qt. Методы работы с сокетами в Qt в сочетании с системой сигналов и слотов позволяет работать с сокетами в асинхронном режиме. Задача реализована с использованием tcp-сокетов в двух программах: «клиент»-версия и «сервер»-версия.

## «Клиент»

В Qt для работы с сокетами используется класс QTcpSocket. При запуске клиента сигналы обьекта класса QTcpSocket привязываются к соответствующим слотам, открывается диалоговое окно, которое позволяет выбрать ip-адрес и порт сервера, к которому необходимо подключиться, после чего происходит попытка подключиться к серверу.

Все запросы и ответы структурированы. Первые 2 байта определяют размер запроса (в байтах), 3 байт определяет действие, которое необходимо сделать (отправить сообщение, передача имени серверу), остальные – содержимое запроса.

После присоединения к серверу клиент отправляет информацию о себе. После этого пользователь может отправлять и принимать сообщения.

## «Сервер»

Для работы tcp-сервера в Qt используется класс QTcpServer. При запуске программы начинается «прослушивание» произвольных ip-адресов, портов, а также включается обработка нового соединения с сервером:

Сервер обрабатывает запросы клиентов, такие как: отправка сообщения всем, отправка приватного сообщения, отправление имени клиента, отключение от сервера. Также сервер может отправлять собственные сообщения всем пользователям и отключать пользователей.

# Листинг

Подключение к серверу:

|  |
| --- |
| connect(sock, SIGNAL(readyRead()), this, SLOT(SockReadyRead()));  connect(sock, SIGNAL(connected()), this, SLOT(SocketConnected()));  connect(sock, SIGNAL(error(QAbstractSocket::SocketError)), this, SLOT(SocketError(QAbstractSocket::SocketError)));  int res = startWin->exec();  qDebug() << res << QDialog::Accepted;  if(res == QDialog::Accepted)  {  startWin->getValues(name, \_host, &\_port);  sock->connectToHost(\_host, \_port);  } |

Формирование и отправка запроса (клиентом или сервером):

|  |
| --- |
| if(sock->state() == QTcpSocket::ConnectedState) {  QByteArray dataBlock;  QDataStream sout(&dataBlock, QIODevice::ReadWrite);  sout << (quint16)(0) << messid << mess;  sout.device()->seek(0);  sout << (quint16)(dataBlock.size() - sizeof(quint16)); messid << name;  sock->write(dataBlock); } |

Клиентский обработчик ответов сервера:

|  |
| --- |
| void MainWindow::SockReadyRead()  {  QDataStream sin(sock);  quint8 command;  QString mess;  QString log = ui->textEdit->toPlainText();  while(sock->bytesAvailable() > 0) {  if(blockSize == 0) {  if(sock->bytesAvailable() < (int)sizeof(quint16))  return;  sin >> blockSize;  }  if(sock->bytesAvailable() < blockSize)  return;  blockSize = 0;  sin >> command >> mess;  switch(command) {  case SERVER\_MESSAGE:  case PUBLIC\_MESSAGE:  ui->textEdit->setPlainText(log + mess + '\n');  break;  case USERS\_LIST:  users = mess.split(",");  users.removeOne("");  ui->listUsers->clear();  ui->listUsers->addItems(users);  ui->listUsers->repaint();  break;  case PRIVATE\_MESSAGE:  QString nickname;  if(!mess.startsWith("["))  return;  nickname = mess.section("]", 0, 0);  nickname.remove(0, 1);  if(!privList.contains(nickname)) {  PrivateWindow \*newWin = new PrivateWindow(nickname, name);  connect(newWin, SIGNAL(sending\_mess(QString)),  this, SLOT(out\_private\_mess(QString)));  connect(this, SIGNAL(in\_private\_mess(QString)),  newWin, SLOT(accepting\_mess(QString)));  privList.insert(nickname, newWin);  }  emit in\_private\_mess(mess);  privList[nickname]->show();  break;  }  }  } |

Прослушивание сервером адресов при запуске:

|  |
| --- |
| server = new QTcpServer(this);  server->listen(QHostAddress::Any, 2000);  ui->listUsers->clear();  connect(server, SIGNAL(newConnection()), this, SLOT(NewConnect())); |

Обработка нового соединения сервером:

|  |
| --- |
| void MainWindow::NewConnect() {  QString hi = "Welcome to chat!";  QTcpSocket \*newSock = server->nextPendingConnection();  CourseClient \*newConnect = new CourseClient;  newConnect->sock = newSock;  newConnect->name = "unknown";  newConnect->blockSize = 0;  clients.append(newConnect);  connect(newConnect->sock,SIGNAL(readyRead()), this, SLOT(receiveMess()));  connect(newConnect->sock, SIGNAL(disconnected()), this, SLOT(socketDisconnected()));  QDataStream sin(clients.last()->sock);  quint8 com;  quint16 size;  sin >> size >> com >> newConnect->name;  QByteArray dataBlock;  QDataStream sout(&dataBlock, QIODevice::ReadWrite);  sout << ((quint16)0) << ((quint8)SERVER\_MESSAGE);  sout << hi;  sout.device()->seek(0);  sout << (quint16)(dataBlock.size() - sizeof(quint16));  newSock->write(dataBlock);  } |

Отключение сервером клиента (“kick”):

|  |
| --- |
| int k;  for(k = 0; k < ui->listUsers->count(); ++k) {  if(ui->listUsers->item(k)->isSelected())  break;  }  if(k == ui->listUsers->count())  return;  sendToUser(clients.at(k), (quint8)(SERVER\_MESSAGE), QString("[SERVER] You are kicked"));  if(clients.at(k)->sock->state() == QTcpSocket::ConnectedState);  clients.at(k)->sock->disconnectFromHost(); |

Обработка сервером запросов клиента:

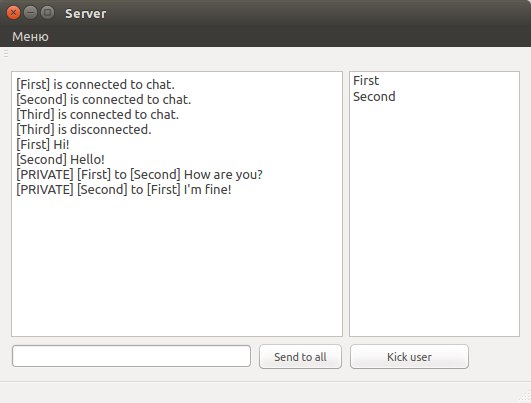
|  |
| --- |
| QTcpSocket \*sock = (QTcpSocket \*)sender();  int senderClient;  for(senderClient = 0; senderClient < clients.length(); ++senderClient) {  if(clients.at(senderClient)->sock == sock)  break;  }  CourseClient \*cl = clients.at(senderClient);  quint8 command;  QString mess;  QString log = ui->textEdit->toPlainText();  receiveFromUser(cl, command, mess);  switch(command) {  case PUBLIC\_MESSAGE: {  for(int j = 0; j < clients.length(); ++j) {  sendToUser(clients.at(j),  (quint8)(PUBLIC\_MESSAGE),  QString("[" + clients.at(senderClient)->name + "] " + mess));  }  ui->textEdit->setPlainText(ui->textEdit->toPlainText() + "[" + clients.at(senderClient)->name + "] " + mess + "\n");  break;  }  case NEW\_CONNECTION: {  QString connMess = "[" + mess + "] is connected to chat.";  ui->textEdit->setPlainText(ui->textEdit->toPlainText() + connMess + "\n");  clients.at(senderClient)->name = mess;  ui->listUsers->addItem(mess);  ui->listUsers->repaint();  QString users\_list;  for(int j = 0; j < clients.length(); ++j) {  users\_list += clients.at(j)->name + ",";  }  for(int j = 0; j < clients.length(); ++j) {  sendToUser(clients.at(j), (quint8)(USERS\_LIST), users\_list);  if(senderClient == j)  continue;  sendToUser(clients.at(j), (quint8)(SERVER\_MESSAGE), connMess);  }  break;  }  case PRIVATE\_MESSAGE: {  QString messName;  if(!mess.startsWith("["))  return;  messName = mess.section("]", 0, 0);  messName.remove(0, 1);  int accepter;  for(accepter = 0; accepter < clients.length(); ++accepter) {  if(clients.at(accepter)->name == messName)  break;  }  sendToUser(clients.at(accepter),  (quint8)(PRIVATE\_MESSAGE),  QString("[" + clients.at(senderClient)->name + "] " + mess.section("]", 1, 1)));  qDebug() << messName << mess.section("]", 1, 1);  ui->textEdit->setPlainText(ui->textEdit->toPlainText() + "[PRIVATE] [" + clients.at(senderClient)->name + "] to " + mess + "\n");  break;  } } |

Отключение клиента от сервера:

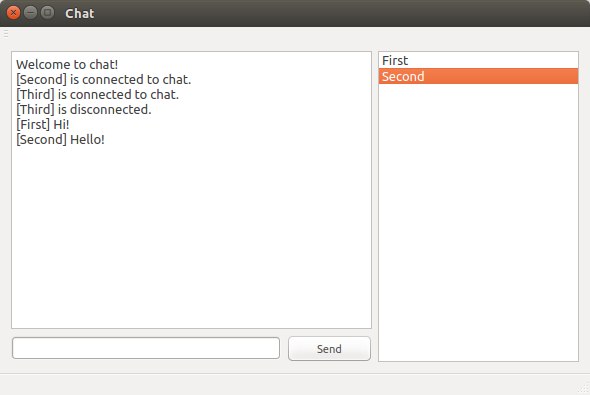
|  |
| --- |
| QTcpSocket \*sock = (QTcpSocket \*)sender();  int i;  for(i = 0; i < clients.length(); ++i) {  if(clients.at(i)->sock == sock)  break;  }  QString deleteName = clients.at(i)->name;  QString serverMess = "[" + deleteName + "] is disconnected.";  clients.removeAt(i);  QString users;  for(int j = 0; j < clients.length(); ++j) {  users += clients.at(j)->name + ",";  }  for(int j = 0; j < clients.length(); ++j) {  sendToUser(clients.at(j), (quint8)(USERS\_LIST), users);  sendToUser(clients.at(j), (quint8)(SERVER\_MESSAGE), serverMess);  } |

# Результаты работы

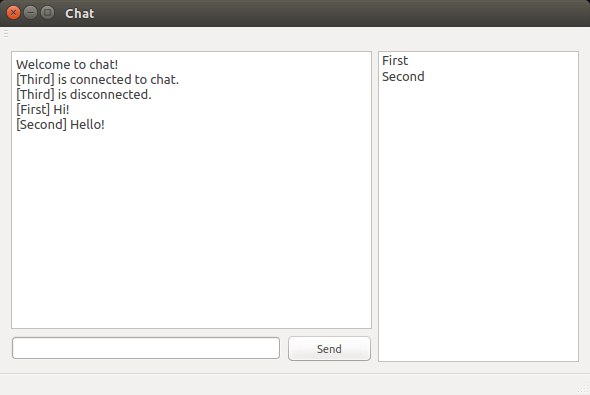
Сервер:



Окно пользователя first:



Окно пользователя Second:



Приватный чат First и Second:

