Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Факультет электронно-информационных систем

Кафедра ИИТ

Лабораторная работа №4

за 5 семестр

По дисциплине: «ОСиСП»

Выполнил:

студент 3 курса

группы ПО-4(1)

Жук В. А.

Проверила:

Дряпко А.В.

2021

Лабораторная работа №4

Цель работы: ознакомиться с возможностями, предлагаемыми Qt для поддержки сетевого взаимодействия программ.

Вариант 9

Задание:

1) Разработать сетевую утилиту для автоматического обновления приложения, разработанного в лабораторных работах 1-3. Утилита может иметь произвольный интерфейс, определяемый ее функциональными особенностями.

2) Программа должна состоять из двух взаимодействующих частей – клиентской, устанавливаемой на компьютере с обновляемым приложением и серверной, выполняющейся на любом компьютере в локальной либо глобальной сети.

3) Клиентская часть осуществляет соединение с сервером и проверку обновлений для приложения. При наличии обновлений, все необходимые файлы загружаются и копируются в директорию с целевым приложением. В противном случае выдается соответствующее сообщение. Обработать возможные исключительные ситуации (отсутствие соединения с сервером).

4) Внести изменения в исходный проект приложения с учетом специфики загружаемых обновлений (например, хранение структуры уровня для игрового приложения в отдельном файле). То есть обновляемые ресурсы должны быть отделены от основного приложения.

5) Обновляемые компоненты по вариантам (DLL из лабораторной работы No3):



**Текст программы:**

1. **main.cpp**

#include <QCoreApplication>

#include <myserver.h>

int **main**(int argc, char \*argv[])

{

QCoreApplication a(*argc*, *argv*);

MyServer Server;

Server.StartServer();

*return* a.exec();

}

1. **myserver.cpp**

#include "myserver.h"

MyServer::**MyServer**(){}

MyServer::~***MyServer***(){}

void MyServer::**StartServer**(){

*if*(*this*->listen(QHostAddress::*Any*, 5555)) {

qDebug() << "Listening";

}

*else* {

qDebug() << "Not Listening";

}

}

void MyServer::**incomingConnection**(int socketDeskriptor) {

socket = *new* QTcpSocket(*this*);

socket->*setSocketDescriptor*(socketDeskriptor);

connect(socket,SIGNAL(readyRead()), *this*, SLOT(sockReady()));

connect(socket,SIGNAL(disconnected()),*this*,SLOT(sockDisc()));

qDebug()<<socketDeskriptor<<"Client connected";

qDebug()<<"Send client connect status - YES";

}

void MyServer::**sockReady**() {

Data = socket->readAll();

qDebug() << "Select from Client" << Data;

*if*(!Data.isEmpty()) {

QDir server\_version(QDir::currentPath());

QStringList filter;

filter << "\*.json";

foreach(QFileInfo info, server\_version.entryInfoList(filter)) {

filter.clear();

filter << info.absoluteFilePath();

}

qDebug() << filter;

QFile file(filter.back());

*if* (!file.*open*(QIODevice::*ReadOnly*))

*return*;

path\_to\_Download = file.readAll();

*if*(QString(Data) == path\_to\_Download) {

qDebug() << "Send to Client" << "Actuale";

socket->write("Actuale");

}

*else* *if*(Data[0] == 'T') {

QStringList client\_data;

client\_data.append(QString(Data).split(" "));

Data.clear();

QDir dir\_server(QDir::currentPath() + "/themes\_interfases");

QStringList find\_filter;

Data.append(dir\_server.path()+' ');

bool ok = dir\_server.exists();

*if* (ok)

{

dir\_server.setFilter(QDir::*Files* | QDir::*Hidden* | QDir::*NoSymLinks*);

dir\_server.setSorting(QDir::*Name*);

QFileInfoList list = dir\_server.entryInfoList();

*for* (int i = 0; i < list.size(); ++i)

{

QFileInfo fileInfo = list.at(i);

find\_filter.append(fileInfo.fileName());

}

}

QStringList sen\_to\_clien;

*for*(int i = client\_data.size(); i < find\_filter.size(); i++) {

sen\_to\_clien.append(find\_filter[i]);

}

foreach (*const* QString &str, sen\_to\_clien)

{

Data.append(str);

*if*(&str != find\_filter.last()) {

Data.append(" ");

}

*else* {

Data.append(" ");

Data.append(path\_to\_Download);

}

}

qDebug() << "Send to Client" << Data;

socket->write(Data);

}

*else* {

qDebug() << "Send to Client" << "Need update";

socket->write("Need update");

}

socket->*waitForBytesWritten*(100);

}

}

void MyServer::**sockDisc**(){

qDebug()<<"Disconnect";

socket->deleteLater();

}

1. **main.h**

#ifndef MYSERVER\_H

#define MYSERVER\_H

#include <QTcpServer>

#include <QTcpSocket>

#include <QDir>

#include <QFile>

*class* **MyServer**: *public* QTcpServer{

Q\_OBJECT

*public*:

**MyServer**();

~***MyServer***();

QTcpSocket \* socket;

QByteArray Data;

QString path\_to\_Download;

*public* slots:

void **StartServer**();

void **incomingConnection**(int socketDeskriptor);

void **sockReady**();

void **sockDisc**();

};

#endif *//* *MYSERVER\_H*

1. **server.pro**

QT += network core

CONFIG += c++11 console

CONFIG -= app\_bundle

DEFINES += QT\_DEPRECATED\_WARNINGS

SOURCES += \

main.cpp \

myserver.cpp

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target

HEADERS += \

myserver.h

1. **mods.cpp**

#include "mode.h"

#include <QDebug>

Mode::**Mode**(QObject \*parent) : QObject(*parent*)

{

qDebug() << name() << "created";

}

Mode::~***Mode***()

{

qDebug() << name() << "destroy";

}

QString Mode::**name**()

{

*return* "mode 2";

}

QFont Mode::***changeView***()

{

QFont palette("Times New Roman", 10, QFont::*Bold*, *true*);

*return* palette;

}

1. **mods.h**

#ifndef MODE\_H

#define MODE\_H

#include <QObject>

#include <interface.h>

*class* **Mode** : *public* QObject, *public* Interface

{

Q\_OBJECT

Q\_PLUGIN\_METADATA(**IID** "calendar.Interface" **FILE** "Interface.json")

Q\_INTERFACES(**Interface**)

*public*:

**Mode**(QObject \*parent =0);

~***Mode***();

QString **name**();

*virtual* QFont ***changeView***();

};

#endif *//* *MODE\_H*

1. **levels.cpp**

#include "levels.h"

QString **countLevels**() {

return QString("2");

}

QString **level1**() {

return QString("10 8");

}

QString **level2**() {

return QString("30 11");

}

1. **mods.pro**

QT += gui

QT += core

TEMPLATE = lib

DEFINES += MODE\_LIBRARY

TARGET = Mode2

CONFIG += plugin

DESTDIR = ../Calendar/release/plugins

INCLUDEPATH += ../interfaces

DEFINES += QT\_DEPRECATED\_WARNINGS

DISTFILES += Interface.json

SOURCES += \

mode.cpp

HEADERS += \

mode.h

unix {

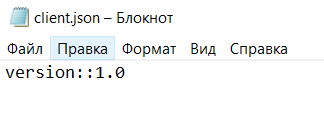
target.path = /usr/lib

}

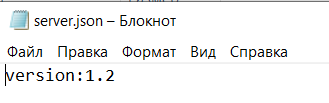
!*isEmpty*(target.path): INSTALLS += target

**Результаты тестирования программы:**

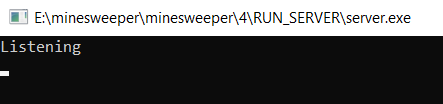
В client.json хранится текущая версия приложения:



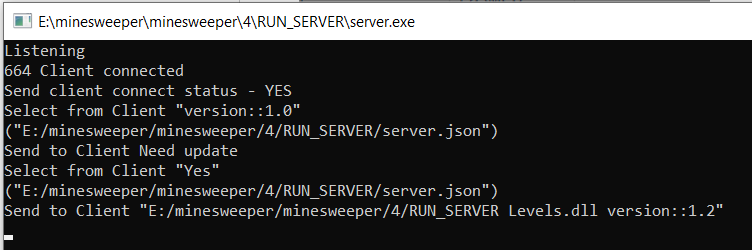
В server.json хранится последняя версия, до которой можно обновить приложение:

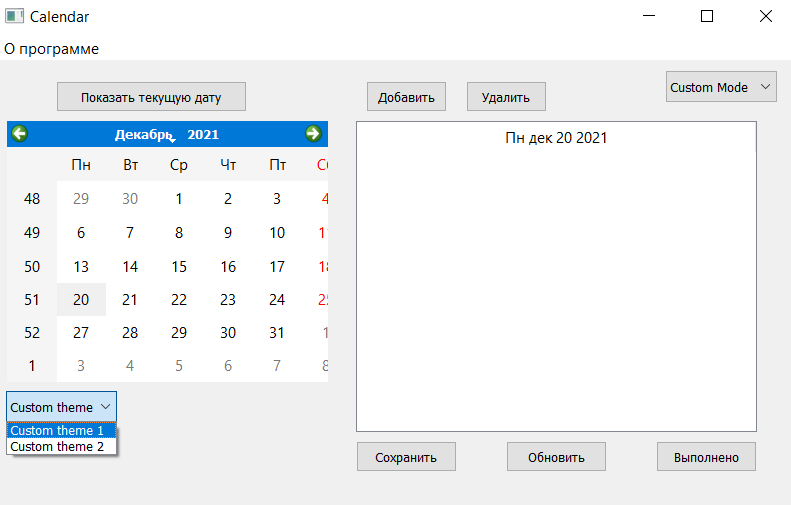
****

Запуск сервера:

****

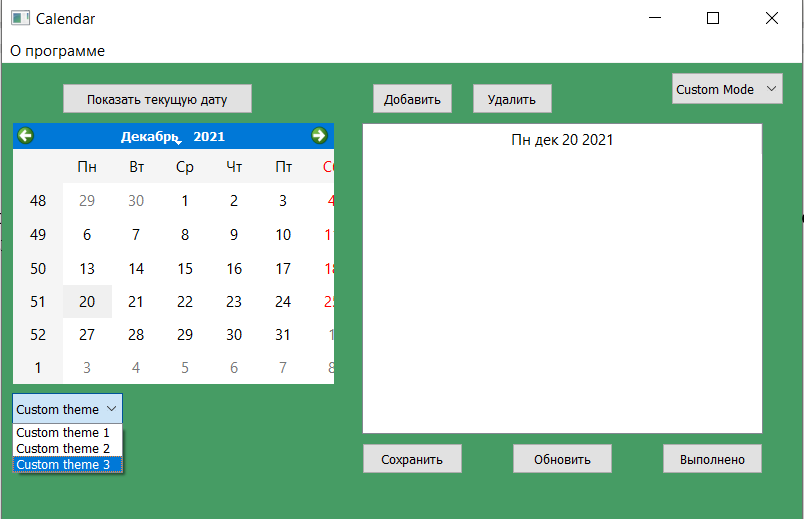
Подключение к серверу и обновление приложения до последней версии:

****



Изображение выглядит как текст

Автоматически созданное описание



Изображение выглядит как текст

Автоматически созданное описание

**Вывод:** Ознакомился с возможностями, предлагаемыми Qt для поддержки сетевого взаимодействия программ.