Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Факультет электронно-информационных систем

Кафедра ИИТ

Лабораторная работа №5

за 5 семестр

По дисциплине: «ОСиСП»

Выполнил:

студент 3 курса

группы ПО-4(1)

Жук В. А.

Проверила:

Дряпко А.В.

2021

Лабораторная работа №5

Цель работы: познакомиться с возможностями, предлагаемыми фреймворком Qt, для разработки многопоточных приложений.

Вариант 9

1) Основное задание заключается в доработке функционала обновления, разработка которого производилась в ЛР No4. Нужно интегрировать указанную функцию в само приложение, без использования стороннего клиента. При этом серверная часть приложения остается без изменений (возможны некоторые доработки сервера, без изменения общей клиент-серверной архитектуры);

2) Проверка обновления должна осуществляться автоматически по таймеру (QTimer) либо по

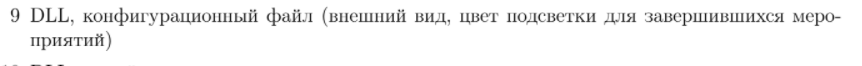
непосредственному запросу пользователя. Предусмотреть выбор из меню политики обновления (с пользовательским подтверждением, без подтверждения/автоматически);

3) Сам процесс обновления должен осуществляться с использованием отдельного потока (QThread) с минимальной вовлечённостью пользователя;

4) Необходимо отображать прогресс обновления (для этого можно использовать строку состояния – QStatusBar);

5) Для демонстрации процесса обновления и независимой работы основного и вспомогательного потоков приложения осуществлять передачу с сервера обновления помимо основных обновляемых компонентов (в соответствии с вариантом задания) одного-двух крупных файлов с произвольным содержимым (например, видео).

6) Обновляемые компоненты по вариантам (ЛР No4):



7) Процесс обновления логируется. При завершении обновления пользователю выдается соответствующее сообщение.

**Текст программы:**

1. **thread.h**

#ifndef THREAD\_H

#define THREAD\_H

#include <QTcpSocket>

#include <QThread>

#include <QTimer>

#include <QDir>

#include <QFile>

*class* **Thread** : *public* QObject

{

Q\_OBJECT

*public*:

*explicit* **Thread**(QString name);

QTcpSocket\* socket;

QByteArray Data;

int progress = 0;

QTimer \*timer;

*public* slots:

void **sockReady**();

void **sockDisc**();

void **run**();

void **share\_video**();

signals:

void **time\_load**(int);

void **load\_update**(int);

void **load\_video**(int);

*private*:

QString name;

QString path\_server;

};

#endif *//* *THREAD\_H*

1. **thread.cpp**

#include "thread.h"

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

Thread::**Thread**(QString s) : name(s) {

}

void Thread::**sockReady**() {

*if*(socket->*waitForConnected*(100)) {

socket->*waitForReadyRead*(100);

Data = socket->readAll();

QStringList rec\_data\_update;

QString new\_version;

rec\_data\_update.append(QString(Data).split(" "));

new\_version.append(rec\_data\_update.last());

path\_server = rec\_data\_update.front();

QDir new\_client\_version(QDir::current());

QStringList filter;

filter << "\*.json";

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

filter.clear();

filter << info.absoluteFilePath();

}

QFile file(filter.back());

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

*return*;

file.write(new\_version.toStdString().data());

*for*(int i = 1; i < rec\_data\_update.size()-1; i++) {

QFile::copy(rec\_data\_update.front()+'/'+rec\_data\_update[i], dir\_client.path()+'/'+rec\_data\_update[i]);

}

*//create* *log*

QDir write\_log(QDir::currentPath() + "/logs");

QStringList formatFile;

formatFile << "\*.txt";

foreach(QFileInfo info, write\_log.entryInfoList(formatFile)) {

formatFile.clear();

formatFile << info.absoluteFilePath();

}

QFile log(formatFile.front());

*if* (!log.*open*(QIODevice::*WriteOnly*))

*return*;

QString text = "Update Modules and version "+ rec\_data\_update.back();

log.write(text.toStdString().data());

}

*//* *sockDisc();*

emit load\_update(1);

share\_video();

}

void Thread::**sockDisc**() {

socket->disconnected();

}

void Thread::**run**() {

socket = *new* QTcpSocket();

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

socket->*connectToHost*("127.0.0.1", 5555);

Data.clear();

QStringList find\_filter;

bool ok = dir\_client.exists();

*if* (ok) {

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

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

QFileInfoList list = dir\_client.entryInfoList();

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

QFileInfo fileInfo = list.at(i);

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

}

qDebug() << find\_filter << endl;

}

foreach (*const* QString &str, find\_filter) {

Data.append(str);

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

Data.append(" ");

}

}

qDebug() << Data << endl; *//get* *list* *themes\_interfases* *client*

socket->write(Data);

socket->*waitForBytesWritten*(100);

emit time\_load(58);

}

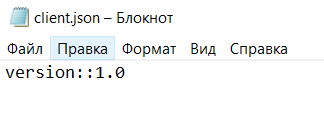
void Thread::**share\_video**() {

*if*(QFile::copy(path\_server+"/video/DCs.Legends.of.Tomorrow.S06E11.1080p.rus.LostFilm.TV.mkv", dir\_client.path() + "/video/DCs.Legends.of.Tomorrow.S06E11.1080p.rus.LostFilm.TV.mkv")) emit load\_video(1);

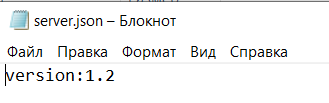
}

**Результаты тестирования программы:**

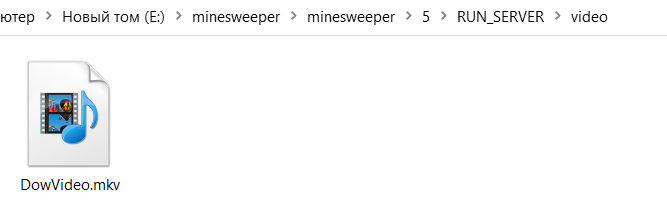
В client.json хранится текущая версия приложения:



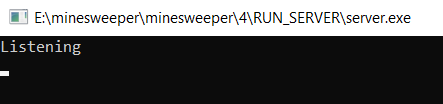
В server.json хранится последняя версия, до которой можно обновить приложение:

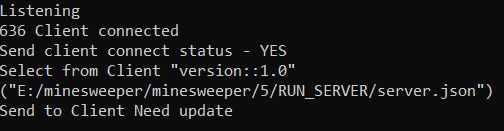
****

Папка на стороне сервера, которое содержит видео:



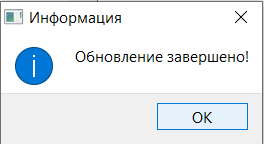
Запуск сервера:

****



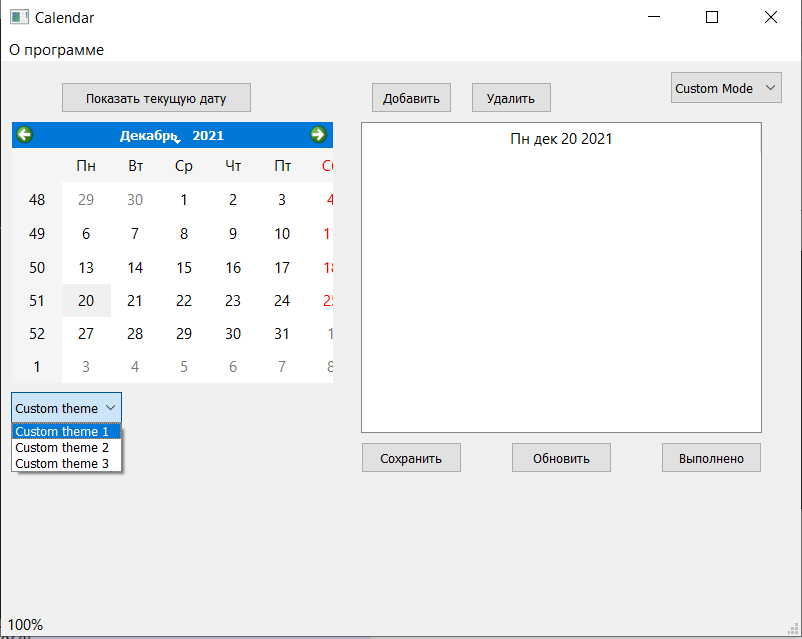
Изображение выглядит как текст

Автоматически созданное описание



Изображение выглядит как текст

Автоматически созданное описание



Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Вывод:** Ознакомился с возможностями, предлагаемыми фреймворком Qt, для разработки многопоточных приложений.