

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Кафедра ИИТ

ЛАБОРАТОРНАЯ РАБОТА №4
По дисциплине ОСиСП за II семестр
«Сетевое взаимодействие программ»

Выполнил:
Студент 3-го курса
Группы ПО-5
Игнатюк А.А.
Проверил:
Дряпко А.В.

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

Вариант 7: Игра Сапер.

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 №3): №7. DLL, конфигурационный файл (количество мин, размер поля).
6. Процесс обновления логируется. При завершении обновления пользователю выдается соответствующее сообщение.
7. Параллельные потоки при выполнении работы не использовать.

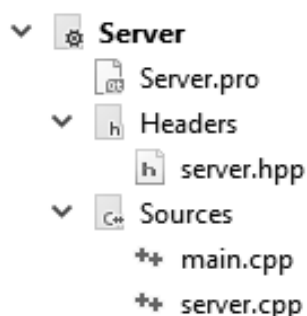


Рисунок 1 - Структура проекта сервера.

```
< > main.cpp <No Symbols>
1  #include <QCoreApplication>
2
3  #include "server.hpp"
4
5  int main(int argc, char *argv[]) {
6      QCoreApplication a(argc, argv);
7      Server v_Server{};
8      return a.exec();
9  }
10
```

Рисунок 2 - Содержимое файла main.cpp.

```

< > server.hpp <Select Symbol>
1  #pragma once
2
3  #ifndef SERVER_HPP
4  #define SERVER_HPP
5
6  #include <QTcpSocket>
7  #include <QTcpServer>
8
9  class Server : public QTcpServer {
10     Q_OBJECT
11
12     public:
13         Server();
14         ~Server();
15
16     private:
17         QTcpSocket* m_SocketPtr{nullptr};
18         const QString mc_Version{"1.2"};
19
20         const QVector<QPair<QString, QStringList>> mc_Changes{
21             {QString{"1.0"}, QStringList{}},
22             {QString{"1.1"}, QStringList{"first_new_file.mp4"}},
23             {QString{"1.2"}, QStringList{"second_new_file.mp4"}}};
24
25         const QString mc_Separator{",,,,"};
26
27         std::uint64_t m_LogCounter{};
28
29     public slots:
30         void incomingConnection(qintptr handle) Q_DECL_OVERRIDE;
31
32         void sf_socket_read();
33         void sf_socket_disconnect();
34     };
35
36 #endif // SERVER_HPP
37

```

Рисунок 3 - Содержимое файла server.hpp.

```

< > server.cpp <Select Symbol>
1  #include "server.hpp"
2
3  #include <QDir>
4  #include <QFile>
5
6  #include <fstream>
7  #include <string>
8
9  Server::Server() {
10     this->m_SocketPtr = new QTcpSocket{};
11
12     this->connect(this->m_SocketPtr, SIGNAL(readyRead()), this,
13                 SLOT(sf_socket_read()));
14     this->connect(this->m_SocketPtr, SIGNAL(disconnected()), this,
15                 SLOT(sf_socket_disconnect()));
16
17     if (this->listen(QHostAddress::Any, 2809)) {
18         qDebug() << "Listening...";
19         return;
20     }
21
22     qDebug() << "Error while listening!";
23 }
24
25 Server::~Server() { delete this->m_SocketPtr; }
26

```

Рисунок 4 - Содержимое файла server.cpp.

Продолжение рисунка 4.

```
27 void Server::incomingConnection(qintptr handle) {
28     this->m_SocketPtr->setSocketDescriptor(handle);
29     qDebug() << "Connected successfully!";
30 }
31
32 void Server::sf_socket_read() {
33     QStringList v_Data{QString{this->m_SocketPtr->readAll()}.split(mc_Separator)};
34
35     const QString c_ClientVersion{v_Data.first()};
36     v_Data.pop_front();
37
38     if (this->mc_Version == c_ClientVersion) {
39         this->m_SocketPtr->write(QByteArray{"0"});
40         qDebug() << "Everything is up to date!";
41         return;
42     }
43
44     qDebug() << "Sending data to client...";
45     this->m_SocketPtr->write(QByteArray{QString{
46         '1' + mc_Separator + mc_Version +
47         mc_Separator}.toString().c_str()});
48
49     auto v_It{this->mc_Changes.begin()};
50
51     for (; v_It != this->mc_Changes.end(); ++v_It) {
52         if (v_It->first == c_ClientVersion) {
53             ++v_It;
54             break;
55         }
56     }
57
58     std::ofstream v_Fout{QString{
59         QDir::currentPath() + "/logs/log_" + QString::number(++m_LogCounter) +
60         ".txt"}.toString()};
61
62     if (!v_Fout.is_open()) {
63         qDebug() << " Can not save log!";
64     }
65
66     std::uint64_t v_ChangeNumber{};
67
68     QByteArray v_DataToSend{};
69
70     for (; v_It != this->mc_Changes.end(); ++v_It) {
71         QDir v_ChangesDir(QDir::currentPath() + "/versions/" + v_It->first);
72         v_ChangesDir.setFilter(QDir::Files | QDir::NoSymLinks);
73
74         if (v_Fout.is_open()) {
75             v_Fout << QString{"Change: " + QString::number(++v_ChangeNumber) +
76                 "\nClient version: " + c_ClientVersion +
77                 "\nServer version: " + mc_Version + "\nFiles to send: "}
78                 .toString();
79         }
80
81         QStringList v_NewFiles{};
82
83         for (const QFileInfo& c_File : v_ChangesDir.entryInfoList()) {
84             v_NewFiles.append(c_File.fileName());
85
86             if (v_Fout.is_open()) {
87                 v_Fout << QString{c_File.fileName() + " "}.toString();
88             }
89         }
90
91         if (v_Fout.is_open()) {
92             v_Fout << "\n\n";
93         }
94
95         for (const QString& c_FileName : v_NewFiles) {
96             v_DataToSend.append(QString{
97                 '2' + mc_Separator + c_FileName +
98                 mc_Separator}.toString()
99                 .c_str());
100 }
```

Продолжение рисунка 4.

```
101         QFile v_File(v_ChangesDir.path() + '/' + c_FileName);
102         constexpr uint64_t c_PART_SIZE{4096};
103
104         if (v_File.open(QIODevice::ReadOnly)) {
105             while (true) {
106                 const QByteArray c_DataPart{v_File.read(c_PART_SIZE)};
107
108                 if (c_DataPart.size() == 0) {
109                     break;
110                 }
111
112                 v_DataToSend.append(c_DataPart);
113             }
114
115             v_File.close();
116         }
117
118         v_DataToSend.append(mc_Separator.toStdString().c_str(),
119                             mc_Separator.size());
120     }
121 }
122
123 if (v_Fout.is_open()) {
124     v_Fout.close();
125 }
126
127 this->m_SocketPtr->write(v_DataToSend);
128 m_SocketPtr->waitForBytesWritten(-1);
129 qDebug() << "Data was sent successfully!";
130 }
131
132 void Server::sf_socket_disconnect() {
133     this->m_SocketPtr->close();
134     qDebug() << "Disconnected successfully!\n";
135 }
136
```

1. Новые поля класса

```
59     QTcpSocket* m_SocketPtr{nullptr};
63     QString m_Version{"1.0"};
64     const QString mc_Separator{",,,,,"};
86     std::uint64_t m_LogCounter{};
```

2. Новые методы класса

```
112     void f_load_difficulty();
113
114     void f_load_version();
115     void f_write_version();
134     void sf_socket_read();
135     void sf_socket_disconnect();
```

Рисунок 5 - Изменения в проекте лабораторной работы №1-3.

```

238 ▼ void Sapper::f_load_difficulty() {
239     QFile v_File(QDir::currentPath() + "/files/difficulty.conf");
240
241 ▼     if (!v_File.open(QIODevice::ReadOnly | QIODevice::ExistingOnly)) {
242         sf_create_game();
243         return;
244     }
245
246     const QString c_Data{v_File.readLine()};
247     v_File.close();
248
249 ▼     if (c_Data == "Easy") {
250         sf_create_easy_game();
251 ▼     } else if (c_Data == "Middle") {
252         sf_create_middle_game();
253 ▼     } else if (c_Data == "Hard") {
254         sf_create_hard_game();
255 ▼     } else {
256         sf_create_game();
257     }
258 }
259
260 ▼ void Sapper::f_load_version() {
261     QFile v_File(QDir::currentPath() + "/version.conf");
262
263 ▼     if (!v_File.open(QIODevice::ReadOnly | QIODevice::ExistingOnly)) {
264         return;
265     }
266
267     const QString c_Data{v_File.readLine()};
268     v_File.close();
269
270     m_Version = c_Data;
271 }
272
273 ▼ void Sapper::f_write_version() {
274     QFile v_File(QDir::currentPath() + "/version.conf");
275
276 ▼     if (!v_File.open(QIODevice::WriteOnly | QIODevice::Truncate)) {
277         return;
278     }
279
280     v_File.write(m_Version.toStdString().c_str(), m_Version.size());
281     v_File.close();
282 }
283
653 void Sapper::sf_socket_disconnect() { m_SocketPtr->close(); }
558 ▼ void Sapper::sf_socket_read() {
559     QMessageBox::information(this, "Update Checker",
560                             "Getting data from the server...");
561
562     m_SocketPtr->waitForReadyRead(100);
563     QByteArray v_RawData{m_SocketPtr->readAll()};
564
565 ▼     std::ofstream v_Fout{QString{
566         QDir::currentPath() + "/logs/log_" + QString::number(++m_LogCounter) +
567         ".txt"}.toStdString()};
568
569 ▼     if (v_Fout.is_open()) {
570         v_Fout << QString{"Client version: " + m_Version + '\n'}.toStdString();
571 ▼     } else {
572         qDebug() << " Can not save log!";
573     }
574

```

Рисунок 6 - Изменения в проекте лабораторной работы №1-3 (Реализация).

Продолжение рисунка 6.

```

575     std::int64_t v_PositionNow{},
576     positionNext{v_RawData.indexOf(",,,,", v_PositionNow)};
577
578     while (v_PositionNow < v_RawData.size()) {
579         QString v_ResponseCode{};
580
581         if (positionNext == -1) {
582             v_ResponseCode =
583                 v_RawData.sliced(v_PositionNow, v_PositionNow + v_RawData.size());
584         } else {
585             v_ResponseCode =
586                 v_RawData.sliced(v_PositionNow, positionNext - v_PositionNow);
587         }
588
589         if (v_ResponseCode == "0") {
590             QMessageBox::information(this, "Update Checker", "You are up to date!");
591             return;
592         }
593
594         v_PositionNow = positionNext + mc_Separator.size();
595         positionNext = v_RawData.indexOf(",,,,", v_PositionNow);
596
597         if (v_ResponseCode == "1") {
598             const QString c_Version{
599
600                 v_RawData.sliced(v_PositionNow, positionNext - v_PositionNow)};
601
602             v_PositionNow = positionNext + mc_Separator.size();
603             positionNext = v_RawData.indexOf(",,,,", v_PositionNow);
604
605             m_Version = c_Version;
606             v_Fout << QString{"Server version: " + m_Version + "\nFiles to send: "}
607                 .toStdString();
608             continue;
609         }
610
611         if (v_ResponseCode == "2") {
612             const QString c_NewFileName{
613                 v_RawData.sliced(v_PositionNow, positionNext - v_PositionNow)};
614
615             if (v_Fout.is_open()) {
616                 v_Fout << QString{c_NewFileName + " "}.toStdString();
617             }
618
619             v_PositionNow = positionNext + mc_Separator.size();
620             positionNext = v_RawData.indexOf(",,,,", v_PositionNow);
621
622             const QByteArray c_Data{
623                 v_RawData.sliced(v_PositionNow, positionNext - v_PositionNow)};
624
625             v_PositionNow = positionNext + mc_Separator.size();
626             positionNext = v_RawData.indexOf(",,,,", v_PositionNow);
627
628             QFile v_File(QDir::currentPath() + "/files/" + c_NewFileName);
629
630             if (v_File.open(QIODevice::WriteOnly)) {
631                 v_File.write(c_Data);
632                 v_File.close();
633             }
634
635             continue;
636         }
637
638         break;
639     }
640
641     if (v_Fout.is_open()) {
642         v_Fout << '\n';
643         v_Fout.close();
644     }
645
646     QMessageBox::information(
647         this, "Update Checker",
648         "Updated successfully!\nPlease restart the program!");
649
650     sf_socket_disconnect();
651 }
652

```

Продолжение рисунка 6.

1. Изменения в конструкторе



```
104     m_SocketPtr = new QTcpSocket(this);
105     connect(m_SocketPtr, SIGNAL(readyRead()), this, SLOT(sf_socket_read()));
106     connect(m_SocketPtr, SIGNAL(disconnected()), this,
107            SLOT(sf_socket_disconnect()));
108
109     f_load_version();
110     f_load_difficulty();
```

2. Изменения в деструкторе

```
137     delete m_SocketPtr;
```

Тестирование:

1. Запуск сервера

Server  
11:30:27: Starting C:\Users\User\Desktop\05\lab_4\build-Server-Desktop_Qt_6_1_2_MinGW_64_bit-Debug\debug\Server.exe ...
Listening...

2. Содержимое папок игры



3. Содержимое папок сервера



4. Запуск игры, проверка обновлений

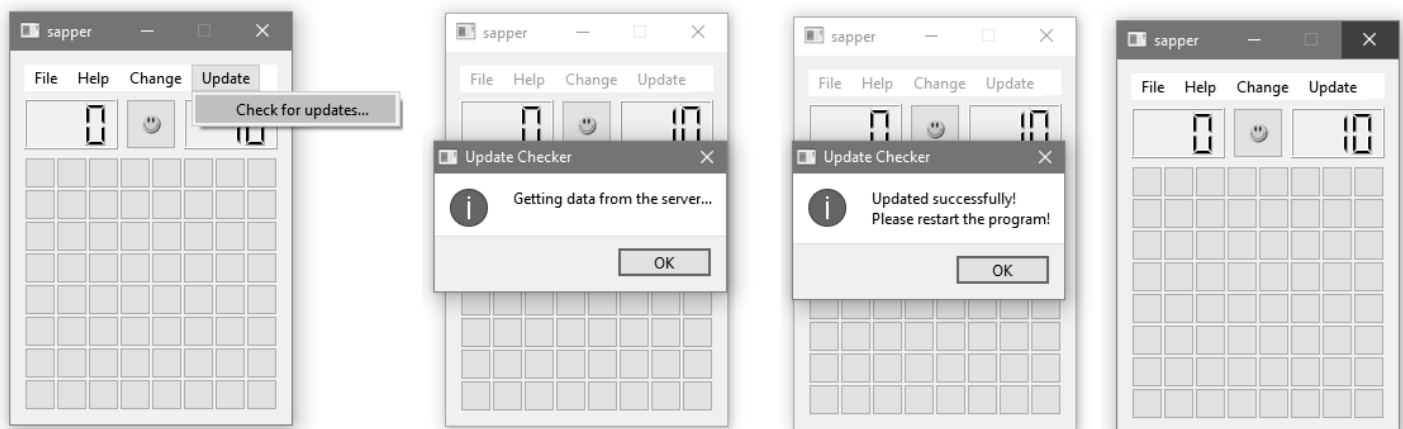


Рисунок 7 - Тестирование программы.

Продолжение рисунка 7.

5. Консольный вывод сервера

```
Server
11:42:56: Starting C:\Users\User\Desktop\05\lab_4\build-Server-Desktop_Qt_6_1_2_MinGW_64_bit-Debug\debug\Server.exe ...
Listening...
Connected successfully!
Sending data to client...
Data was sent successfully!
Disconnected successfully!
```

6. Содержимое папок игры

Корневая папка

- debug
- files
- logs
- release
- .qmake.stash
- Makefile
- Makefile.Debug
- Makefile.Release
- scores.txt
- version.conf

Папка files

- about.dll
- difficulty.conf
- first_new_file.rar
- second_new_file.rar

Папка logs

- log_1.txt

Файл log_1.txt

```
log_1.txt - Notepad
File Edit Format View Help
Client version: 1.0
Server version: 1.2
Files to receive: difficulty.conf first_new_file.rar about.dll difficulty.conf second_new_file.rar
```

7. Содержимое папок сервера

Папка logs

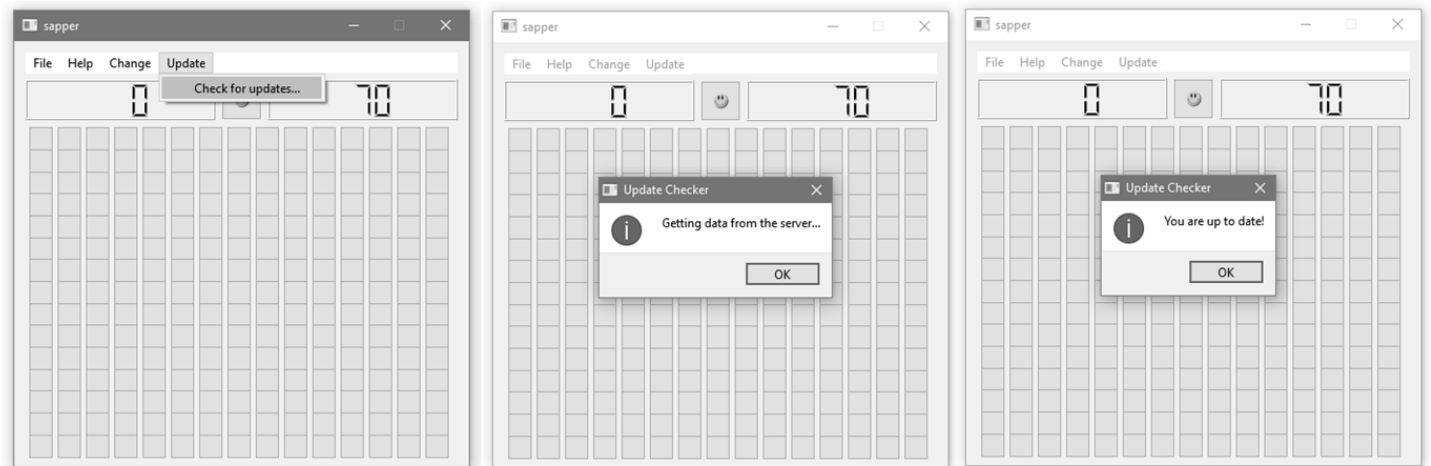
- log_1.txt

Файл log_1.txt

```
log_1.txt - Notepad
File Edit Format View Help
Change: 1
Client version: 1.0
Server version: 1.2
Files to send: difficulty.conf first_new_file.rar

Change: 2
Client version: 1.0
Server version: 1.2
Files to send: about.dll difficulty.conf second_new_file.rar
```

8. Запуск обновленной игры, проверка обновлений



9. Консольный вывод сервера

```
Server
11:42:56: Starting C:\Users\User\Desktop\05\lab_4\build-Server-Desktop_Qt_6_1_2_MinGW_64_bit-Debug\debug\Server.exe ...
Listening...
Connected successfully!
Sending data to client...
Data was sent successfully!
Disconnected successfully!

Connected successfully!
Everything is up to date!
```

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