

Сети ЭВМ и телекоммуникации

В. Е. Бушин

25 декабря 2015 г.

Глава 1

Задание

Разработать приложение–клиент и приложение–сервер платежной системы. Участники платежной системы имеют электронные кошельки. Электронный кошелек имеет уникальный номер. При регистрации пользователя в платежной системе на его счет зачисляется определенная сумма. Пользователя платежной системы могут осуществлять платежи друг другу через приложение–сервер.

1.1 Функциональные требования

Серверное приложение должно реализовывать следующие функции:

1. Прослушивание определенного порта
2. Обработка запросов на подключение по этому порту от клиентов платежной системы
3. Поддержка одновременной работы нескольких клиентов платежной системы через механизм нитей
4. Прием запросов от клиента на регистрацию пользователя, передачу списка электронных кошельков пользователей платежной системы, осуществление платежей одного пользователя другому, проверка состояния счета кошелька
5. Осуществление добавления пользователя в платежную систему, хранение и изменение состояния электронных кошельков в зависимости от платежей пользователей

6. Передача запросов на платежи от одного пользователя другому, подтверждений платежей, номера нового кошелька при регистрации пользователя, списка электронных кошельков
7. Обработка запроса на отключение клиента
8. Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1. Установление соединения с сервером
2. Передача запросов на передачу списка электронных кошельков пользователей платежной системы, платежи одного пользователя другому, проверку состояния счета кошелька
3. Получение от сервера запросов на платеж от другого пользователя, результатов платежа
4. Разрыв соединения
5. Обработка ситуации отключения клиента сервером

Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени сервера платежной системы и номера порта, используемого сервером.

1.2 Нефункциональные требования

При подключении клиента происходит процесс идентификации с помощью логина или процесс регистрации нового пользователя. Пользователь не может вводить команды (кроме команд, связанных с идентификацией), пока идентификация не будет успешной. Один и тот же пользователь одновременно может быть идентифицирован с нескольких клиентских приложений. Серверное и клиентское приложения должны работать на разных ОС.

1.3 Накладываемые ограничения

Серверное приложение не поддерживает одновременное подключение более 100 клиентов. Максимальная длина вводимых пользователем строчковых данных — 255 символов.

Глава 2

Реализация для работы по протоколу TSP

2.1 Прикладной протокол

Команды, доступные клиентскому приложению приведены в таблице 2.1.

Имя команды	Аргумент 1	Аргумент 2
register	новый логин	-
sign in	логин	-
show users	-	-
check wallet	-	-
transfer	имя пользователя	кол-во денег
quit	-	-

Таблица 2.1: Команды, доступные клиентскому приложению

Описание команд:

1. register — создание нового пользователя. Возможные ответы сервера:
 - «ok» — команда выполнена успешно;
 - «not ok» — пользователь с таким именем уже существует.
2. sign in — вход уже существующего пользователя. Возможные ответы сервера:
 - «ok» — команда выполнена успешно;

- «no match» — пользователя с таким именем не найдено.
3. show users — запрос на передачу списка электронных кошельков пользователей платежной системы. В ответ сервер посылает список пользователей платёжной системы, в нём содержится имена пользователей и их идентификаторы.
 4. check wallet — запрос на проверку состояния электронного кошелька. Команда доступна только тестировщикам. В ответ сервер посылает количество денег на счету пользователя.
 5. transfer — запрос на платёж другому пользователю. Команда вводится в 2 этапа:
 - сначала пользователь вводит саму команду;
 - затем вводит имя пользователя кому будет проведён перевод и количество денег для перевода.

Возможные ответы сервера:

- «ok» — команда выполнена успешно;
 - «no match» — пользователя с таким именем не найдено.
6. quit — выход из клиентского приложения.

2.2 Архитектура приложения

При подключении нового клиента сервер создает новый поток. Сначала происходит идентификация подключившегося пользователя. После идентификации начинается процесс работы. Клиент отправляет серверу различные команды, сервер выполняет эти команды и отправляет клиенту результаты выполнения команд.

На сервере используются структура user, в которой содержится вся нужная информация о подключившемся клиенте. После идентификации вся информация о подключившемся клиенте заносится в эту структуру.

Данные о пользователе хранятся в 2-х файлах:

- us.txt — в этом файле хранятся имена пользователей и их идентификаторы, разделённые табуляцией, в одной строчке содержится информация только об одном пользователе;

- money.txt — здесь хранятся идентификаторы пользователей и количество денег на их счету, структура записей такая же как и в файле us.txt.

Пользователю на сервере доступны 2 команды:

1. show — вывод списка подключенных клиентов с номерами их сокетов;
2. disconnect номер клиента — отключение указанного клиента от сервера;

2.3 Тестирование

2.3.1 Описание тестового стенда и методики тестирования

Серверное приложение реализовано в ОС Linux. Тестирование проводилось на ОС Debian 8.1. Сервер запускался на виртуальной машине с подключением к сети через сетевой мост. Клиентское приложение реализовано в ОС Windows. Тестирование проводилось на ОС Windows 10. Для тестирования приложений запускается сервер платежной системы и несколько клиентов. В процессе тестирования проверяются основные возможности приложений по передаче и приему сообщений.

2.3.2 Тестовый план и результаты тестирования

Процесс тестирования:

1. Проверка идентификации:
 - Попытка войти с неправильным логином;
 - Попытка войти с правильным логином;
 - Попытка зарегистрироваться с уже занятым логином;
 - Попытка зарегистрироваться со свободным.

Результаты с попытками входа приведены на рисунке 1, результаты с попытками регистрации приведены на рис.2

```
C:\WINDOWS\system32\cmd.exe
Sign in or register
sign in
Enter your username:
ajgdkgdjadkgasd
No such username. Type correct username or register
Sign in or register
Sign in or register
sign in
Enter your username:
wladex
Hello wladex, you has successfully logged in
Enter the command: Undefined command
Enter the command:
```

Рис.1

```
C:\WINDOWS\system32\cmd.exe
Sign in or register
register
Create new username:
wladex
User with this username is already existing. Create other username
Sign in or register
Sign in or register
register
Create new username:
dima
Hello dima, you has successfully registered and logged in
Enter the command: Undefined command
Enter the command:
```

Рис.2

2. Проверка корректности выполнения всех команд клиента:

- Проверка функции передачи списка электронных кошельков пользователей платежной системы(рис. 3);

```
C:\WINDOWS\system32\cmd.exe
Enter the command: show users
wladex 1
azat 2
lera 3
egor 4
dima 5
Enter the command: _
```

Рис.3

- Проверка функции проверки состояния электронного кошелька (рис. 4);

```
Enter the command: check wallet
6000
Enter the command:
```

Рис.4

- Проверка функции платежа другому пользователю (рис. 5);

```

C:\WINDOWS\system32\cmd.exe
Sign in or register
sign in
Enter your username:
dima
Hello dima, you has successfully logged in
Enter the command: Undefined command
Enter the command: check wallet
6500
Enter the command: transfer
To who and how much do you want transfer money?
wladex 5000
Operation done
Enter the command: Undefined command
Enter the command:
Enter the command: check wallet
5500
Enter the command: check wallet
10500
Enter the command:

```

Рис.5

- Попытка выхода из клиентского приложения (рис. 6);

```

Enter the command: check wallet
10500
Enter the command: quit
Для продолжения нажмите любую клавишу . . .

```

Рис.6

- Проверка корректной работы нескольких клиентов (рис. 7);

```

C:\WINDOWS\system32\cmd.exe
Sign in or register
sign in
Enter your username:
wladex
Hello wladex, you has successfully logged in
Enter the command: Undefined command
Enter the command: show users
wladex 1
azat 2
lera 3
egor 4
dima 5
Enter the command: check wallet
10500
Enter the command: transfer
To who and how much do you want transfer money?
azat 3000
Operation done
Enter the command: Undefined command
Enter the command:

```

```

C:\WINDOWS\system32\cmd.exe
Sign in or register
sign in
Enter your username:
azat
Hello azat, you has successfully logged in
Enter the command: Undefined command
Enter the command: check wallet
7000
Enter the command: check wallet
10000
Enter the command: show users
wladex 1
azat 2
lera 3
egor 4
dima 5
Enter the command:

```

Рис.7

- Проверка отключения клиентов от сервера (рис. 8).

```

lera 3
egor 4
dima 5

show
5 : wladez
4 : lera
disconnect 4
wladez 1
azat 2
lera 3
egor 4
dima 5

show
5 : wladez
disconnect 5

```

C:\WINDOWS\system32\cmd.exe

```

Sign in or register
sign in
Enter your username:
wladez
Hello wladez, you has successfully logined
Enter the command: Undefined command
Enter the command: show users
wladez 1
azat 2
lera 3
egor 4
dima 5

Enter the command: check wallet
Disconnected from server
Для продолжения нажмите любую клавишу . . .

```

C:\WINDOWS\system32\cmd.exe

```

Sign in or register
sign in
Enter your username:
lera
Hello lera, you has successfully logined
Enter the command: Undefined command
Enter the command: show users
wladez 1
azat 2
lera 3
egor 4
dima 5

Enter the command: show users
Disconnected from server
Для продолжения нажмите любую клавишу . . .

```

Рис.8

Глава 3

Реализация для работы по протоколу UDP

3.1 Прикладной протокол

Прикладной протокол UDP-приложения совпадает с протоколом TCP-приложения, описанного ранее в пункте 2.1.

3.2 Архитектура приложения

Архитектура UDP-приложения совпадает с архитектурой TCP-приложения, которая была описана в пункте 2.2. Небольшие изменения коснулись только клиентского приложения. Была добавлена структура `Uclient`, которая была нужна для того, чтобы в ней хранились сокет и структура `sockaddr` с адресом сервера.

3.3 Тестирование

3.3.1 Описание тестового стенда и методики тестирования

Тестовый стенд был описан ранее в пункте 2.3.1

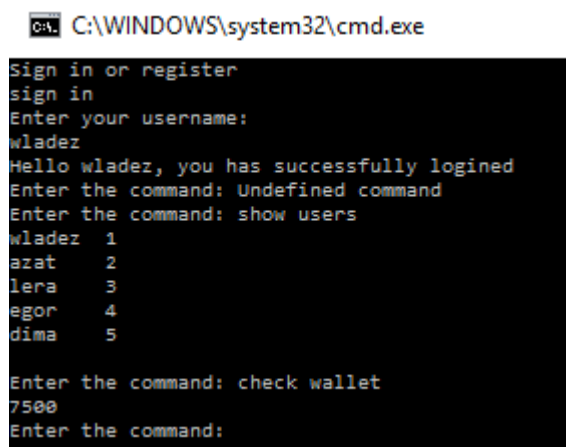
3.3.2 Тестовый план и результаты тестирования

Первый этап тестирования совпадает с тестированием TCP-приложения (пункт 2.3.2). Результаты тестирования совпали с тестированием TCP-

приложения.

Далее UDP реализация приложения была протестирована на устойчивость помехам в сети.

1. Введение дополнительной задержки в сеть 150 ± 50 мс (рис. 9)

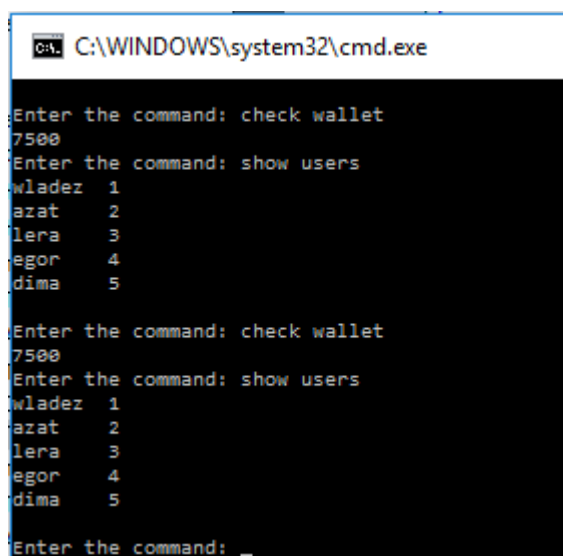


```
C:\WINDOWS\system32\cmd.exe
Sign in or register
sign in
Enter your username:
wladex
Hello wladex, you has successfully logined
Enter the command: Undefined command
Enter the command: show users
wladex 1
azat 2
lera 3
egor 4
dima 5
Enter the command: check wallet
7500
Enter the command:
```

Рис.9

При такой маленькой задержке проблем в работе приложения не возникнет.

2. Введение задержки в сеть 1500 ± 500 мс (рис. 10)

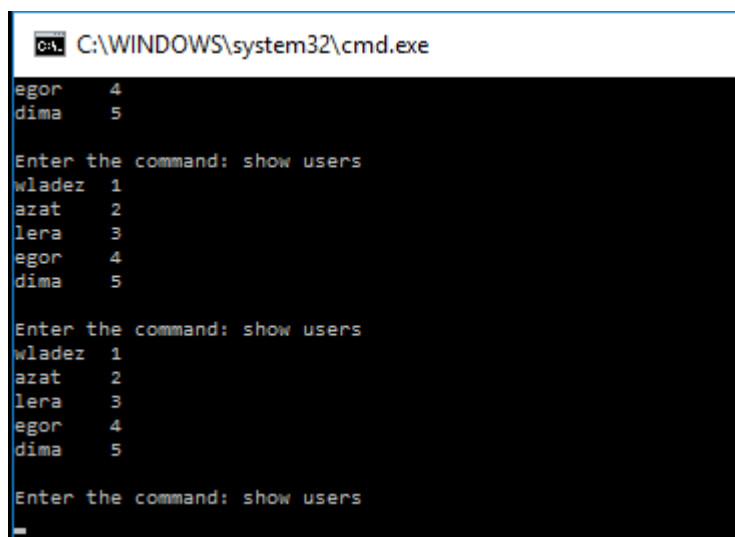


```
C:\WINDOWS\system32\cmd.exe
Enter the command: check wallet
7500
Enter the command: show users
wladex 1
azat 2
lera 3
egor 4
dima 5
Enter the command: check wallet
7500
Enter the command: show users
wladex 1
azat 2
lera 3
egor 4
dima 5
Enter the command: _
```

Рис.10

При большой задержке клиент получает некоторых сообщения от сервера, но они приходят не сразу и команды выполняются за 2 секунды.

3. Введение потери 30% пакетов (рис. 11)



```
C:\WINDOWS\system32\cmd.exe
egor 4
dima 5

Enter the command: show users
wladex 1
azat 2
lera 3
egor 4
dima 5


Enter the command: show users
wladex 1
azat 2
lera 3
egor 4
dima 5

Enter the command: show users
```

Рис.11

При таком проценте потери пакетов приложение практически сразу завершает работу: ответ от сервера теряется в сети и приложение зависает.

4. Введение дубликации 20% пакетов (рис. 12)



```
C:\WINDOWS\system32\cmd.exe
dima 5

Enter the command: show users
wladex 1
azat 2
lera 3
egor 4
dima 5

Enter the command: check wallet
wladex 1
azat 2
lera 3
egor 4
dima 5

Enter the command: check wallet
7500
Enter the command: show users
7500
Enter the command:
```

Рис.12

При таком проценте дубликации пакетов приложение продолжает функционировать, однако в его работе возникают неполадки, на некоторые команды данные отображаются по предыдущей команде.

5. Введение искажения 10% пакетов (рис. 13)

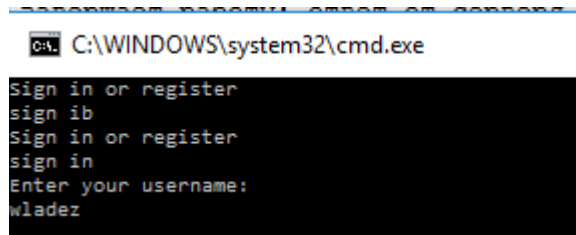


Рис.13

При таком проценте искажения пакетов приложение продолжает функционировать, однако есть большая вероятность того, что оно зависнет.

6. Введение задержки 150 ± 50 мс, искажения 5%, потери 5%, дубликации 5% пакетов (рис. 14)

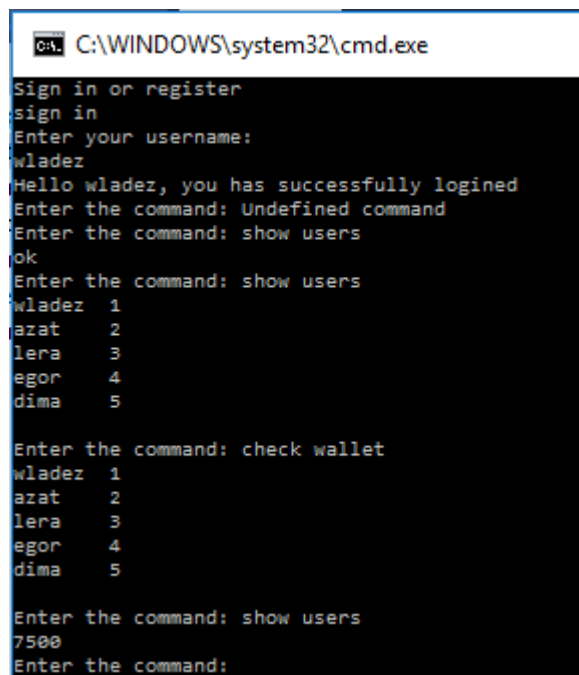


Рис.14

Приложение продолжает функционировать, однако могут возникать неполадки, так же приложение может зависнуть.

В целом можно сделать вывод, что приложение не готово к работе в реальной сети. При возникновении небольших помех в сети приложение продолжит корректно функционировать. Но при потере пакетов или резком понижении качества канала хотя бы по одному параметру приложение быстро зависнет свою работу из-за ошибки.

Глава 4

Выводы

В результате работы был создан прикладной протокол взаимодействия клиент-серверного приложения. В соответствии с прикладным протоколом было создано две реализации приложения для протоколов TCP и UDP. Клиентское и серверное приложения были реализованы для двух разных платформ: ОС Windows и Linux. При реализации использовались стандартные сокеты. Реализации сокетов для использованных ОС идентичны, портирование программ с одной платформы на другую выполняется достаточно просто.

В результате работы была создана клиент-серверная платёжная система. Система состоит из сервера, хранящего сведения о клиентах и состоянии их кошельков. Все поставленные требования были выполнены. Тестирование программ прошло успешно, тесты показали корректность работы приложения в условиях локальной сети.

4.1 Реализация для TCP

Протокол TCP удобен для реализации пользовательских приложений, так как обеспечивает установление соединения и надёжную доставку пакетов. Протокол обеспечивает стабильное надёжное соединение, поэтому при реализации своего протокола не требуется волноваться об этом. Однако, эти дополнительные средства синхронизации требуют больше времени на доставку, т.е. скорость передачи данных ниже чем в UDP.

С помощью утилиты `tc` при тестировании TCP реализации были про- симулированы помехи в сети. Тестирование показало, что приложение надёжно работает при низком и среднем уровне помех.

4.2 Реализация для UDP

Протокол UDP удобен для реализации приложений, не требующих точной доставки пакетов. Он позволяет передавать данные с большей скоростью, однако вероятность потери пакета при этом выше, чем в TCP. Поэтому, использовать данный протокол для реализации поставленной задачи не очень удобно. Требуется использовать дополнительные инструменты для подтверждения корректной доставки, т.е. каким-то образом «симулировать» TCP. Это неудобно и неэффективно. Для обеспечения более надежной доставки при использовании UDP было добавлено подтверждение доставки при обмене между клиентом и сервером.

С помощью утилиты `tc` при тестировании UDP реализации были про-симулированы помехи в сети. Тестирование показало, что при низком уровне помех приложение продолжает функционировать, однако могут появляться ошибки критические для приложения. Если в сети присутствуют сильные помехи, приложение начинает работать некорректно и может зависнуть с большой вероятностью.

Приложения

Описание среды разработки

Серверное приложение реализовывалось в ОС Debian версии 8.1. Среда разработки — Qt Creator версии 5.5.

Клиентское приложения реализовывалось в ОС Windows 10. Среда разработки — Microsoft Visual Studio 2010.

Листинги

ТСР сервер

Файл main.cpp

```
1 #include <QCoreApplication>
2 #include <pthread.h>
3 #include <stdio.h>
4 #include <cstdio>
5 #include <stdlib.h>
6 #include <netdb.h>
7 #include <netinet/in.h>
8 #include <string.h>
9 #include <string>
10 #include <unistd.h>
11 #include "server.h"
12 // #include "user.h"
13
14 int main(int argc, char *argv[])
15 {
16     int sock, newsock, port_num, n;
17     struct sockaddr_in serv_addr, cli_addr;
18     socklen_t clilen;
19     pid_t pid;
20
21     sock=socket(AF_INET, SOCK_STREAM, 0);
22
```

```

23     if (sock < 0)
24     {
25         perror("ERROR opening socket");
26         exit(1);
27     }
28     bzero((char*)&serv_addr, sizeof(serv_addr));
29     port_num=12345;
30
31     serv_addr.sin_family=AF_INET;
32     serv_addr.sin_addr.s_addr=INADDR_ANY;
33     serv_addr.sin_port=htons(port_num);
34
35     if(bind(sock,(struct sockaddr*)&serv_addr,sizeof(
36         serv_addr))<0){
37         perror("ERROR on binding");
38         exit(1);
39     }
40
41     listen(sock,10);
42     cli_len=sizeof(cli_addr);
43     printf("TCP Server Waiting for client on port 12345\n");
44
45     pthread_t serv_thread;
46     if (pthread_create(&serv_thread, NULL, server_handler,
47         NULL) != 0) {
48         printf("Error while creating thread for server\n");
49     }
50
51     while (1) {
52         pthread_t thread;
53         newsock = accept(sock, (struct sockaddr *) &cli_addr
54             , &cli_len);
55
56         if (newsock < 0) {
57             perror("ERROR on accept");
58             close(sock);
59             exit(1);
60         }
61         pid=pthread_create(&thread, NULL, doprocessing, &
62             newsock);
63         if (pid!=0){
64             printf("Error while creating new thread\n");
65             close(newsock);
66             break;
67         }
68     } /* end of while */
69     return 0;
70

```

67 }

Файл server.cpp

```
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <string.h>
4 #include <cstdlib>
5 #include <netdb.h>
6 #include <netinet/in.h>
7 #include <unistd.h>
8 #include "server.h"
9 #include "user.h"
10
11 int clientsCount=0;
12 user connected[maxClients];
13
14 int authentication(int sock){
15     //user connected;
16     char buf[bufSize+1];
17     char new_client[]="new";
18     char exist_client[]="exist";
19     char numb[bufSize];
20     int n,check;
21     int res;
22     strcpy(numb,"\t");
23     bzero(buf, bufSize+1);
24     n=recv(sock, buf, bufSize, 0);//приём сообщения от клиент
25     а, в котором указано
26     //будет ли подключен новый пользователь или уже зарегистр
27     ированный
28     if (n < 0) {
29         perror("ERROR_reading_from_socket");
30         exit(1);
31     }
32     if(strncmp(buf,exist_client,sizeof(exist_client)-1) == 0)
33     {
34         //уже существующий пользователь
35         bzero(buf,bufSize);
36         n=recv(sock, buf, bufSize, 0);
37         if (n < 0) {
38             perror("ERROR_reading_from_socket");
39             pthread_exit(0);
40         }
41         check=check_user(buf);//проверка совпадает ли имя кот
42         орое ввёл пользователь
43         //с именем в файле с пользователями
44         if (check < 0) //нет совпадений
45             n = send(sock,"no_match",sizeof(buf), 0);//говори
46             м клиенту, что нет совпадений
```

```

42         if (n < 0)
43         {
44             perror("ERROR_writing_to_socket");
45             pthread_exit(0);
46         }
47         res=-1;
48     }
49     else{//есть совпадения
50         n = send(sock,"ok",bufSize, 0);//говорим клиенту,
           что всё ок
51         if (n < 0)
52         {
53             perror("ERROR_writing_to_socket");
54             pthread_exit(0);
55         }
56         connected[clientsCount].sock=sock;
57         strcpy(connected[clientsCount].name,buf);
58         connected[clientsCount].uid=check;
59         connected[clientsCount].money=get_money(connected
           [clientsCount].uid);
60         res=1;
61         clientsCount++;
62         printf("Connected_client_%s\n",buf);
63     }
64 }
65 else if(strncmp(buf,new_client,sizeof(new_client)-1) ==
0){
66     //создание нового пользователя
67     bzero(buf,bufSize);
68     n=recv(sock, buf, bufSize, 0);
69     if (n < 0) {
70         perror("ERROR_reading_from_socket");
71         pthread_exit(0);
72     }
73
74     check=check_user(buf);//проверка нет ли уже такого им
           ени у кого-нибудь
75     if(check>0){//есть совпадения
76         n = send(sock,"not_ok",6, 0);//говорим клиенту, ч
           то не ок
77         if (n < 0)
78         {
79             perror("ERROR_writing_to_socket");
80             pthread_exit(0);
81         }
82         res=-1;
83     }
84     else{//нет совпадений, клиент зарегистрирован

```

```

85         n = send(sock, "ok", 2, 0); //говорим клиенту, что
            ещё ок
86         if (n < 0)
87         {
88             perror("ERROR_writing_to_socket");
89             pthread_exit(0);
90         }
91         connected[clientsCount].sock=sock;
92         strcpy(connected[clientsCount].name, buf);
93         connected[clientsCount].money=6000;
94         printf("Connected_client_%s\n", buf);
95         connected[clientsCount].uid=set_newid(); //получен
            ие id пользователя
96         FILE *file;
97         char *fname = "/home/user/project_t/us.txt";
98         file = fopen(fname, "a");
99         fprintf(file, "%s\t", connected[clientsCount].name)
            ; //запись в файл нового пользователя
100        fprintf(file, "%i\n", connected[clientsCount].uid);
101        fclose(file);
102        FILE *mon;
103        char *mon_name="/home/user/project_t/money.txt";
104        mon=fopen(mon_name, "a");
105        fprintf(mon, "%i\t", connected[clientsCount].uid);
106        fprintf(mon, "%i\n", connected[clientsCount].money);
107        fclose(mon);
108        clientsCount++;
109        res=1;
110    }
111 }
112 return res;
113 }
114
115 void* doprocessing (void* newsock) {
116     int n, socket;
117     int aut;
118     char request[bufSize];
119     char buffer[bufSize+1];
120     char command[]="show_users";
121     char quit[]="quit";
122     bzero(request, bufSize);
123     bzero(buffer, bufSize+1);
124     int *tmp=(int*)newsock;
125     socket=*tmp;
126     do {
127         aut = authentication(socket); //процесс аутентификации
            клиента
128     } while (aut < 0);

```

```

129     while(recv(socket, request, bufSize, 0)>=0){
130         //n=recv(socket, request, bufSize, 0);
131         //     if(n<0){
132         //         perror("ERROR reading from socket");
133         //         break;
134         //     }
135         if(strncmp(request,quit,sizeof(quit)-1) == 0){
136             disconnect(socket);
137             break;
138         }
139         else if(strncmp(request,command,sizeof(command)-1) ==
140             0){
141             show_users(socket);
142         }
143         else if(strcmp(request,"wallet") == 0){
144             check_wallet(socket);
145         }
146         else if(strcmp(request,"transf") == 0){
147             transfer(socket);
148         }
149         bzero(request,bufSize);
150     }
151     close(socket);
152 }
153 void show_users(int sock){
154     int n;
155     char tmp[bufSize];
156     char buffer[bufSize+1];
157     bzero(buffer,bufSize+1);
158     FILE *file;
159     char *fname = "/home/user/project_t/us.txt";
160     file = fopen(fname,"r");
161     if(file == NULL)
162     {
163         perror("ERROR on opening file with users");
164         pthread_exit(0);
165     }
166     while (fgets (tmp, sizeof(tmp), file) != NULL){
167         strncat(buffer,tmp,35);
168         printf("%s", tmp);
169     }
170     printf("\n");
171     fclose(file);
172     n = send(sock,buffer,sizeof(buffer), 0);
173     if (n < 0)
174     {
175         pthread_exit(0);
176     }

```

```

177 }
178
179 void check_wallet(int sock){
180     int n,uid;
181     char buffer[bufSize+1];
182     bzero(buffer,bufSize+1);
183     int i = 0;
184     int j=0;
185     for (i = 0; i <= clientsCount; ++i) {
186         if (connected[i].sock == sock){
187             uid=connected[i].uid;
188             connected[i].money=get_money(uid);
189             j=i;
190         }
191     }
192     sprintf(buffer,"%i", connected[j].money);
193     n = send(sock,buffer,sizeof(buffer), 0);
194     if (n < 0)
195     {
196         pthread_exit(0);
197     }
198 }
199
200 void transfer(int sock){
201     int n,value,money;
202     int dest=0;
203     char buffer[bufSize+1];
204     bzero(buffer,bufSize+1);
205     n=recv(sock, buffer, bufSize+1, 0);
206     if(n<0){
207         perror("ERROR reading from socket");
208         pthread_exit(0);
209     }
210     printf("%s\n",buffer);
211     char *tmp=strstr(buffer,"_");
212     value=atoi(tmp);
213     strcpy(tmp,"\0");
214     dest=check_user(buffer);
215     if (dest < 0) { //нет совпадений
216         n = send(sock,"no_match",8, 0); //говорим клиенту, что
            нет совпадений
217         if (n < 0)
218         {
219             pthread_exit(0);
220         }
221     }
222     else{
223         money=get_money(dest);
224         money+=value;

```

```

225         set_money(dest, money);
226         int i, j;
227         for (i = 0; i <= clientsCount; ++i) {
228             if (connected[i].sock == sock){
229                 dest=connected[i].uid;
230                 connected[i].money=get_money(dest);
231                 connected[i].money-=value;
232                 j=i;
233             }
234         }
235         set_money(dest, connected[j].money);
236         n = send(sock, "ok", 2, 0);
237         if (n < 0)
238         {
239             pthread_exit(0);
240         }
241     }
242 }
243
244 int check_user(char buf[]){
245     char name[bufSize+1];
246     char tmp[bufSize+1];
247     char id[10];
248     int res=-1;
249     int k;
250     FILE *file;
251     char *fname = "/home/user/project_t/us.txt";
252     file = fopen(fname, "r");
253     bzero(name, bufSize+1);
254     strcpy(name, buf);
255     if(file == NULL)
256     {
257         perror("ERROR on opening file with users");
258         exit(1);
259     }
260     int i = 0;
261     bzero(tmp, bufSize+1);
262     while(fscanf(file, "%s", tmp) != EOF){
263         //fscanf(file, "%s", tmp);
264         if(k==0){
265             strcpy(id, tmp);
266             res=atoi(id);
267             printf("%d\n", res);
268             break;
269         }
270         if(!(i%2)){
271             k=strcmp(name, tmp);
272             printf("%s\n", tmp);
273         }

```



```

274         i++;
275     }
276     fclose(file);
277     return res;
278 }
279
280 void disconnect(int sock){
281     int i = 0;
282     for (i = 0; i < clientsCount; ++i) {
283         if (connected[i].sock == sock)
284             break;
285     }
286     if (i != clientsCount) {
287         for (++i; i < clientsCount; ++i) {
288             connected[i - 1] = connected[i];
289         }
290         --clientsCount;
291     }
292 }
293
294 void* server_handler(void*){
295     while (1) {
296         char command[bufSize];
297         bzero(command, bufSize);
298         scanf("%s", command);
299         if (strcmp(command, "show") == 0) {
300             for (int i = 0; i < clientsCount; ++i) {
301                 printf("%d: %s\n", connected[i].sock,
302                     connected[i].name);
303             }
304             } else if (strcmp(command, "disconnect") == 0) {
305                 int sock = 0;
306                 scanf("%d", &sock);
307                 disconnect(sock);
308                 int n = send(sock, "exit", 4, 0);
309                 if (n < 0)
310                 {
311                     perror("ERROR_writing_to_socket");
312                     exit(1);
313                 }
314                 close(sock);
315             } else {
316                 printf("Undefined command\n");
317             }
318     }
}

```

Файл server.h

```
1 #ifndef SERVER
```

```

2 #define SERVER
3 void *doprocessing(void *sock);
4 void show_users(int sock);
5 int authentication(int sock);
6 int check_user(char buf[]);
7 void check_wallet(int sock);
8 void transfer(int sock);
9 void *server_handler(void *);
10 void disconnect(int sock);
11 #endif // SERVER

```

Файл user.cpp

```

1 #include <stdio.h>
2 #include <string.h>
3 #include <cstdlib>
4 #include "user.h"
5
6 int get_money(int usid){//узнать количество денег, имеющееся
   у пользователя
7     int i=0;
8     int res,tmp=0;
9     int k=-1;
10    char buffer[bufSize+1];
11    FILE *file;
12    char *fname="/home/user/project_t/money.txt";
13    file = fopen(fname,"r");
14    if(file == NULL)
15    {
16        perror("ERROR_on_openning_file_with_money");
17        exit(1);
18    }
19    bzero(buffer,bufSize+1);
20    while(!feof(file)){
21        fscanf(file,"%s", buffer);
22        tmp=atoi(buffer);
23        if(k==0){
24            res=tmp;
25            break;
26        }
27        if(!(i%2)){
28            if(usid==tmp)
29                k=0;
30        }
31        i++;
32    }
33    fclose(file);
34    return res;
35 }
36

```

```

37 int set_money(int uid, int value){
38     char buf[15];
39     int tmp=0;
40     int before=0;
41     int after=0;
42     int spaces=0;
43     fpos_t pos;
44     FILE *file;
45     char *fname="/home/user/project_t/money.txt";
46     file = fopen(fname,"r+");
47     if(file == NULL)
48     {
49         perror("ERROR on opening file with money");
50         exit(1);
51     }
52     int k=1;
53     int i = 0;
54     while(fscanf(file,"%s", buf)!=EOF){
55         if(k==0){
56             before=ftell(file);
57             fsetpos(file,&pos);
58             fprintf(file,"\t%i",value);
59             after=ftell(file);
60             if(before>after) {
61                 spaces += before - after;
62                 while(spaces!=0){
63                     fputc('_', file);
64                     spaces--;
65                 }
66             }
67             fflush(file);
68             break;
69         }
70         fgetpos(file, &pos);
71         if(!(i%2)){
72             tmp=atoi(buf);
73             if(tmp==uid){
74                 k=0;
75             }
76         }
77         i++;
78     }
79 }
80 }
81
82 int set_newid(){//задание нового id пользователя при регистра
    чуу
83     int res=0;
84     char uid[15];

```

```

85     FILE *file;
86     char *fname = "/home/user/project_t/us.txt";
87     file = fopen(fname, "r");
88     bzero(uid, sizeof(uid));
89     if(file == NULL)
90     {
91         perror("ERROR_on_openning_file_with_users");
92         exit(1);
93     }
94     int i = 0;
95     while(fscanf(file, "%s", uid) != EOF){
96         if(i%2){
97             res=atoi(uid);
98         }
99         i++;
100     }
101     fclose(file);
102     return res+1;
103 }

```

Файл user.h

```

1  #ifndef USER
2  #define USER
3  #define bufSize 255
4  #define maxClients 100
5  typedef struct{
6      char name[bufSize];
7      int sock;
8      int uid;
9      int money;
10 }user;
11
12 //user();
13 int set_newid();
14 int get_money(int usid);
15 int set_money(int uid, int value);
16 #endif // USER

```

ТСР клиент

Файл client.cpp

```

1  //client
2  #include <winsock2.h>
3  #include <ws2tcpip.h>
4  #include <stdlib.h>
5  #include <stdio.h>

```

```

6 #include <string.h>
7
8 #pragma comment (lib, "Ws2_32.lib")
9 #pragma comment (lib, "Mswsock.lib")
10 #pragma comment (lib, "AdvApi32.lib")
11
12 #define bufSize 255
13
14 int authentication(int sockfd);
15 void showUsers(int sockfd, char command[]);
16 void checkWallet(int sockfd);
17 void transfer(int sockfd);
18 int disconnect(int sockfd, char* buf);
19
20 int main(int argc, char *argv[]) {
21     WORD wVersionRequested = MAKEWORD(1, 1);           // Stuff
22     WSADATA wsaData;
23     int sockfd, portno, n;
24     int aut;
25     struct sockaddr_in serv_addr;
26     struct hostent *server;
27     char quit[]="quit";
28     char show[]="show_users";
29     char wallet[]="check_wallet";
30     char transf[]="transfer";
31     char buffer[bufSize+1];
32
33     WSASStartup(wVersionRequested, &wsaData);
34     //portno=12345;
35     if (argc < 3) {
36         fprintf(stderr, "usage_%s_hostname_port\n", argv[0]);
37         exit(0);
38     }
39
40     portno = atoi(argv[2]);
41
42     /* Create a socket point */
43     sockfd = socket(AF_INET, SOCK_STREAM, 0);
44
45     if (sockfd < 0) {
46         perror("ERROR_opening_socket");
47         exit(1);
48     }
49
50     server = gethostbyname(argv[1]);
51     if (server == NULL) {
52         fprintf(stderr, "ERROR, no such host\n");
53         exit(0);

```

```

54     }
55
56     memset((char *) &serv_addr, 0, sizeof(serv_addr));
57     serv_addr.sin_family = AF_INET;
58     serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
59     //strncpy((char *)server->h_addr, (char *)&serv_addr.
        sin_addr.s_addr, server->h_length);
60     serv_addr.sin_port = htons(portno);
61
62     /* Now connect to the server */
63     if (connect(sockfd, (struct sockaddr*)&serv_addr, sizeof(
        serv_addr)) < 0) {
64         perror("ERROR connecting");
65         exit(1);
66     }
67
68     /* Now ask for a message from the user, this message
69        * will be read by server
70     */
71     do {
72         aut = authentication(sockfd); //процесс аутентифик
            ации клиента
73     } while (aut < 0);
74     while (1){
75         printf("Enter the command: ");
76         memset(buffer, 0, bufSize+1);
77         fgets(buffer, bufSize+1, stdin);
78
79         if(strncmp(buffer, quit, sizeof(quit)-1) == 0){
80             n = send(sockfd, buffer, strlen(buffer), 0);
81             if (n < 0) {
82                 perror("ERROR writing to socket");
83                 exit(1);
84             }
85             closesocket(sockfd);
86             break;
87         }
88         else if(strncmp(buffer, show, sizeof(show)-1) == 0){
89             showUsers(sockfd, buffer);
90         }
91         else if(strncmp(buffer, wallet, sizeof(wallet)-1) == 0){
92             checkWallet(sockfd);
93         }
94         else if(strncmp(buffer, transf, sizeof(transf)-1) == 0){
95             transfer(sockfd);
96         }
97         else{
98             printf("Undefined command\n");
99         }

```

```

100     }
101     return 0;
102 }
103
104 void showUsers(int sockfd, char command[]){
105     int n;
106     char buffer[bufSize+1];
107     memset(buffer, 0, bufSize);
108     strcpy(buffer, command);
109     n = send(sockfd, buffer, strlen(buffer), 0);
110     if (n < 0) {
111         perror("ERROR_writing_to_socket");
112         closesocket(sockfd);
113         exit(1);
114     }
115
116     /* Now read server response */
117     memset(buffer, 0, bufSize);
118     n = recv(sockfd, buffer, bufSize+1, 0);
119     if (n < 0) {
120         perror("ERROR_reading_from_socket");
121         closesocket(sockfd);
122         exit(1);
123     }
124     disconnect(sockfd, buffer);
125     printf("%s\n", buffer);
126 }
127
128 void checkWallet(int sockfd){
129     int n;
130     char buffer[bufSize+1];
131     n=send(sockfd, "wallet", 6, 0);
132     if (n < 0) {
133         perror("ERROR_writing_to_socket");
134         closesocket(sockfd);
135         exit(1);
136     }
137     memset(buffer, 0, bufSize+1);
138     n = recv(sockfd, buffer, bufSize+1, 0);
139     if (n < 0) {
140         perror("ERROR_reading_from_socket");
141         closesocket(sockfd);
142         exit(1);
143     }
144     disconnect(sockfd, buffer);
145     printf("%s\n", buffer);
146 }
147
148 void transfer(int sockfd){

```

```

149     int n;
150     char tmp[bufSize];
151     char buffer[bufSize+1];
152     memset(buffer, 0, bufSize+1);
153     n=send(sockfd, "transf", 6, 0);
154     if (n < 0) {
155         perror("ERROR_writing_to_socket");
156         closesocket(sockfd);
157         exit(1);
158     }
159     printf("To_who_and_how_much_do_you_want_transfer_money?\n");
160     scanf("%s",buffer);
161     scanf("%s",tmp);
162     strcat(buffer,"_");
163     strcat(buffer,tmp);
164     n=send(sockfd, buffer, bufSize+1, 0);
165     memset(buffer, 0, bufSize+1);
166     n=recv(sockfd, buffer, bufSize+1,0);
167     if (n < 0) {
168         perror("ERROR_reading_from_socket");
169         closesocket(sockfd);
170         exit(1);
171     }
172     disconnect(sockfd, buffer);
173     if(strcmp(buffer,"no_match")==0){
174         printf("There_is_no_user_with_such_username\n");
175     }
176     else if(strcmp(buffer,"ok")==0){
177         printf("Operation_done\n");
178     }
179     else{
180         printf("Some_error_occurs_during_the_operation\n");
181     }
182 }
183
184 int disconnect(int sockfd, char* buf){
185     if (strcmp(buf, "exit") == 0) {
186         printf("Disconnected_from_server\n");
187         closesocket(sockfd);
188         exit(1);
189     }
190     else return -1;
191 }
192
193 int authentication(int sockfd){
194     char login[bufSize +1];
195     char reg[bufSize +1];
196     char buffer[bufSize +1];

```



```

197 char ok[]="ok";
198 char sign[]="sign_in";
199 char registration[]="register";
200 int n;
201 int res=-1;
202 memset(buffer, 0, bufSize+1);
203 printf("Sign_in_or_register\n");
204 fgets(buffer, bufSize+1, stdin);
205 //scanf("%s", &buffer);
206 if(strncmp(buffer, sign, sizeof(sign)-1) == 0){//вход сущес
    твующего пользователя
207     printf("Enter_your_username:\n");
208     scanf("%s", &login);
209     n=send(sockfd, "exist", bufSize, 0); //посылка серверу со
        общения о том, что входит существующий пользовател
        ь
210     if (n < 0) {
211         perror("ERROR_writing_to_socket");
212         closesocket(sockfd);
213         exit(1);
214     }
215     n=send(sockfd, login, bufSize, 0); //посылка серверу имен
        и пользователя
216     if (n < 0) {
217         perror("ERROR_writing_to_socket");
218         closesocket(sockfd);
219         exit(1);
220     }
221     memset(buffer, 0, bufSize+1);
222     n = recv(sockfd, buffer, bufSize+1, 0); //ответ от серв
        ера, правильны ли данные или нет
223     if (n < 0) {
224         perror("ERROR_reading_from_socket");
225         closesocket(sockfd);
226         exit(1);
227     }
228     disconnect(sockfd, buffer);
229     if(strncmp(buffer, ok, sizeof(ok)-1) == 0){//данные пра
        вильны
230         printf("Hello%s, you has successfully logged\n"
            , login);
231         res=1;
232     }
233     else{//данные не правильны
234         printf("No such username. Type correct username_
            or_register\n");
235         res=-1;
236     }
237 }

```

```

238     else if(strncmp(buffer,registration,sizeof(registration)
239         -1) == 0){//регистрация нового пользователя
240         printf("Create_new_username:\n");
241         scanf("%s", &reg);
242         n=send(sockfd,"new",bufSize,0);//отправка сообщения се
243         рверу о регистрации нового пользователя
244         if (n < 0) {
245             perror("ERROR_writing_to_socket");
246             closesocket(sockfd);
247             exit(1);
248         }
249         n=send(sockfd,reg,bufSize,0);//отправка имени нового п
250         ользователя
251         if (n < 0) {
252             perror("ERROR_writing_to_socket");
253             closesocket(sockfd);
254             exit(1);
255         }
256         memset(buffer, 0, bufSize+1);
257         n = recv(sockfd, buffer, bufSize+1,0);//ответ от серв
258         ера, правильны ли данные или нет
259         if (n < 0) {
260             perror("ERROR_reading_from_socket");
261             closesocket(sockfd);
262             exit(1);
263         }
264         disconnect(sockfd, buffer);
265         if(strncmp(buffer,ok,sizeof(ok)-1) == 0){//данные пра
266         вильны
267         printf("Hello%s, you has successfully registered
268             and logged\n",reg);
269         res=1;
270         }
271         else{//данные не правильны
272         printf("User with this username is already
273             existing. Create other username\n");
274         }
275     }
276     return res;
277 }

```

UDP сервер

Файл main.cpp

```

1 #include <QCoreApplication>
2 #include <pthread.h>
3 #include <stdio.h>

```

```

4 #include <stdio>
5 #include <stdlib.h>
6 #include <netdb.h>
7 #include <netinet/in.h>
8 #include <string.h>
9 #include <string>
10 #include <unistd.h>
11 #include "server.h"
12
13
14 int main(int argc, char *argv[])
15 {
16     int sock, port_num;
17     struct sockaddr_in serv_addr;
18     socklen_t clilen;
19     pid_t pid;
20
21     sock=socket(AF_INET, SOCK_DGRAM, 0);
22
23     if (sock < 0)
24     {
25         perror("ERROR opening socket");
26         exit(1);
27     }
28     bzero((char*)&serv_addr, sizeof(serv_addr));
29     //port_num=12345;
30     port_num = atoi(argv[1]);
31
32     serv_addr.sin_family=AF_INET;
33     serv_addr.sin_addr.s_addr=INADDR_ANY;
34     serv_addr.sin_port=htons(port_num);
35
36     if(bind(sock,(struct sockaddr*)&serv_addr,sizeof(
37         serv_addr))<0){
38         perror("ERROR on binding");
39         exit(1);
40     }
41
42     printf("UDP Server Waiting for client on port 12345\n");
43
44     pthread_t serv_thread;
45     if (pthread_create(&serv_thread, NULL, server_handler,
46         NULL) != 0) {
47         printf("Error while creating thread for server\n");
48     }
49
50     while (1) {
51         pthread_t thread;
52         user client = new_connection(sock);

```

```

51         pid=pthread_create(&thread, NULL, doprocessing, (void
           *)&client);
52         if (pid!=0){
53             printf("Error_while_creating_new_thread\n");
54             //break;
55         }
56     } /* end of while */
57     return 0;
58 }
59 }

```

Файл server.cpp

```

1  #include <stdio.h>
2  #include <pthread.h>
3  #include <string.h>
4  #include <cstdlib>
5  #include <netdb.h>
6  #include <netinet/in.h>
7  #include <unistd.h>
8  #include "server.h"
9
10 int clientsCount=0;
11 user connected[maxClients];
12
13 user new_connection(int sockfd){
14     struct sockaddr_in clientaddr; /* client addr */
15     int clientlen = sizeof(clientaddr); /* byte size of
        client's address */
16     char buf[bufSize];
17     int err = 0;
18
19     bzero(buf, bufSize);
20     err = recvfrom(sockfd, buf, bufSize, 0, (struct sockaddr
        *) &clientaddr, (unsigned int*)&clientlen);
21     if (err < 0) {
22         perror("ERROR_reading_from_socket");
23         exit(1);
24     }
25
26     int newsockfd;
27     uint16_t newport;
28     new_socket(&newsockfd, &newport);
29     bzero(buf, bufSize);
30     sprintf(buf, "%d", newport);
31     err = sendto(sockfd, buf, strlen(buf), 0, (struct
        sockaddr *) &clientaddr, clientlen);
32     if (err < 0) {
33         perror("ERROR_writing_to_socket");
34         exit(1);

```

```

35     }
36
37     // create new socket for this client
38     struct sockaddr_in newclientaddr; /* client addr */
39     int newclientlen = sizeof(newclientaddr); /* byte size of
        client's address */
40     bzero(buf, bufSize);
41     err = recvfrom(newsockfd, buf, bufSize, 0, (struct
        sockaddr *) &newclientaddr, (unsigned int*)&
        newclientlen);
42     if (err < 0) {
43         perror("ERROR_reading_from_socket");
44         exit(1);
45     }
46
47     user client;
48     client.cli_addr=newclientaddr;
49     client.clilen=newclientlen;
50     client.sock=newsockfd;
51     return client;
52 }
53
54 void new_socket(int* sockfd, uint16_t* port){
55     struct sockaddr_in serveraddr;
56     int err = 0;
57     *sockfd = socket(AF_INET, SOCK_DGRAM, 0);
58     *port = 12345 + (clientsCount + 1);
59     if(*sockfd<0){
60         perror("ERROR_opening_socket");
61         exit(1);
62     }
63     //int optval = 1;
64     //setsockopt(*sockfd, SOL_SOCKET, SO_REUSEADDR, (const
        void *)&optval , sizeof(int));
65
66     bzero((char *) &serveraddr, sizeof(serveraddr));
67     serveraddr.sin_family = AF_INET;
68     serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
69     serveraddr.sin_port = htons((unsigned short)*port);
70
71     err = bind(*sockfd, (struct sockaddr *) &serveraddr,
        sizeof(serveraddr));
72     if(err<0){
73         perror("ERROR_on_binding");
74         exit(1);
75     }
76 }
77
78 int authentication(user client){

```

```

79 char buf[bufSize+1];
80 char new_client[]="new";
81 char exist_client[]="exist";
82 char numb[bufSize];
83 int n,check;
84 int res;
85 strcpy(numb, "\t");
86 bzero(buf, bufSize+1);
87 n=recvfrom(client.sock, buf, bufSize, 0, (struct sockaddr
      *)&client.cli_addr, &client.clilen);//приём сообщения
      от клиента, в котором указано
88 //будет ли подключен новый пользователь или уже зарегистр
      ированный
89 if (n < 0) {
90     perror("ERROR_reading_from_socket");
91     exit(1);
92 }
93 if(strncmp(buf,exist_client,sizeof(exist_client)-1) == 0)
    {
94     //уже существующий пользователь
95     bzero(buf,bufSize);
96     n=recvfrom(client.sock, buf, bufSize, 0, (struct
      sockaddr *) &client.cli_addr, &client.clilen);
97     if (n < 0) {
98         perror("ERROR_reading_from_socket");
99         pthread_exit(0);
100     }
101     check=check_user(buf);//проверка совпадает ли имя кот
      орое ввёл пользователь
102     //с именем в файле с пользователями
103     if (check < 0) //нет совпадений
104     {
105         n = sendto(client.sock,"no_match",sizeof(buf), 0,
      (struct sockaddr *) &client.cli_addr, client.
      clilen);//говорим клиенту, что нет совпадений
106         if (n < 0)
107         {
108             perror("ERROR_writing_to_socket");
109             pthread_exit(0);
110         }
111         res=-1;
112     }
113     else//есть совпадения
114     {
115         n = sendto(client.sock,"ok",bufSize, 0, (struct
      sockaddr *) &client.cli_addr, client.clilen);
      //говорим клиенту, что всё ок
116         if (n < 0)
117         {
118             perror("ERROR_writing_to_socket");
119             pthread_exit(0);

```

```

118         }
119         strcpy(client.name, buf);
120         client.uid = check;
121         client.money = get_money(client.uid);
122         res = 1;
123         add(client);
124         printf("Connected_client_%s\n", buf);
125     }
126 }
127 else if(strncmp(buf, new_client, sizeof(new_client)-1) ==
0){
128     //создание нового пользователя
129     bzero(buf, bufSize);
130     n = recvfrom(client.sock, buf, bufSize, 0, (struct
sockaddr *) &client.cli_addr, &client.clilen);
131     if (n < 0) {
132         perror("ERROR_reading_from_socket");
133         pthread_exit(0);
134     }
135
136     check = check_user(buf); //проверка нет ли уже такого им
ени у кого-нибудь
137     if(check > 0){ //есть совпадения
138         n = sendto(client.sock, "not_ok", 6, 0, (struct
sockaddr *) &client.cli_addr, client.clilen);
        //говорим клиенту, что не ок
139         if (n < 0)
140         {
141             perror("ERROR_writing_to_socket");
142             pthread_exit(0);
143         }
144         res = -1;
145     }
146     else{ //нет совпадений, клиент зарегистрирован
147         n = sendto(client.sock, "ok", 2, 0, (struct
sockaddr *) &client.cli_addr, client.clilen);
        //говорим клиенту, что всё ок
148         if (n < 0)
149         {
150             perror("ERROR_writing_to_socket");
151             pthread_exit(0);
152         }
153         strcpy(client.name, buf);
154         client.money = 6000;
155         printf("Connected_client_%s\n", buf);
156         client.uid = set_newid(); //получение id пользователя
я
157         FILE *file;
158         char *fname = "/home/user/project_t/us.txt";

```

```

159         file = fopen(fname,"a");
160         fprintf(file,"%s\t",client.name);//запись в файл
161         нового пользователя
162         fprintf(file,"%i\n",client.uid);
163         fclose(file);
164         FILE *mon;
165         char *mon_name="/home/user/project_t/money.txt";
166         mon=fopen(mon_name,"a");
167         fprintf(mon,"%i\t",client.uid);
168         fprintf(mon,"%i\n",client.money);
169         fclose(mon);
170         add(client);
171         res=1;
172     }
173     return res;
174 }
175
176 void* doprocessing (void* c) {
177     user* cli = (user*) c;
178     user client = *cli;
179     int aut;
180     char request[bufSize];
181     char buffer[bufSize+1];
182     char command[]="show_users";
183     char quit[]="quit";
184     bzero(request,bufSize);
185     bzero(buffer,bufSize+1);
186     do {
187         aut = authentication(client); //процесс аутентификации
188         клиента
189         while (aut < 0);
190         while(recvfrom(client.sock, request, bufSize, 0, (struct
191             sockaddr *) &client.cli_addr, &client.clilen)>=0){
192             //n=recv(socket, request, bufSize, 0);
193             if(n<0){
194                 perror("ERROR reading from socket");
195                 break;
196             }
197             if(strncmp(request,quit,sizeof(quit)-1) == 0){
198                 disconnect(client.sock);
199                 break;
200             }
201             else if(strncmp(request,command,sizeof(command)-1) ==
202                 0){
203                 show_users(client);
204             }
205             else if(strcmp(request,"wallet") == 0){
206                 check_wallet(client);
207             }
208         }
209     } while(1);
210 }

```



```

204     }
205     else if(strcmp(request,"transf") == 0){
206         transfer(client);
207     }
208     bzero(request,bufSize);
209 }
210 close(client.sock);
211 }
212
213 void show_users(user client){
214     int n;
215     char tmp[bufSize];
216     char buffer[bufSize+1];
217     bzero(buffer,bufSize+1);
218     FILE *file;
219     char *fname = "/home/user/project_t/us.txt";
220     file = fopen(fname,"r");
221     if(file == NULL)
222     {
223         perror("ERROR on opening file with users");
224         pthread_exit(0);
225     }
226     while (fgets (tmp, sizeof(tmp), file) != NULL){
227         strncat(buffer,tmp,35);
228         printf("%s", tmp);
229     }
230     printf("\n");
231     fclose(file);
232     n = sendto(client.sock,buffer,sizeof(buffer), 0, (struct
        sockaddr *) &client.cli_addr, client.clilen);
233     if (n < 0)
234     {
235         pthread_exit(0);
236     }
237 }
238
239 void check_wallet(user client){
240     int n,uid;
241     char buffer[bufSize+1];
242     bzero(buffer,bufSize+1);
243     int i = 0;
244     int j=0;
245     for (i = 0; i <= clientsCount; ++i) {
246         if (connected[i].sock == client.sock){
247             uid=connected[i].uid;
248             connected[i].money=get_money(uid);
249             j=i;
250         }
251     }

```

```

252     sprintf(buffer,"%i", connected[j].money);
253     n = sendto(client.sock,buffer,sizeof(buffer), 0, (struct
        sockaddr *) &client.cli_addr, client.clilen);
254     if (n < 0)
255     {
256         pthread_exit(0);
257     }
258 }
259
260 void transfer(user client){
261     int n,value,money;
262     int dest=0;
263     char buffer[bufSize+1];
264     bzero(buffer,bufSize+1);
265     n=recvfrom(client.sock, buffer, bufSize+1, 0, (struct
        sockaddr *) &client.cli_addr, &client.clilen);
266     if(n<0){
267         perror("ERROR_reading_from_socket");
268         pthread_exit(0);
269     }
270     printf("%s\n",buffer);
271     char *tmp=strstr(buffer,"_");
272     value=atoi(tmp);
273     strcpy(tmp,"\0");
274     dest=check_user(buffer);
275     if (dest < 0) {//нет совпадений
276         n = sendto(client.sock,"no_match",8, 0, (struct
            sockaddr *) &client.cli_addr, client.clilen);//сообщим клиенту, что нет совпадений
277         if (n < 0)
278         {
279             pthread_exit(0);
280         }
281     }
282     else{
283         money=get_money(dest);
284         money+=value;
285         set_money(dest,money);
286         int i,j;
287         for (i = 0; i <= clientsCount; ++i) {
288             if (connected[i].sock == client.sock){
289                 connected[i].money-=value;
290                 dest=connected[i].uid;
291                 j=i;
292             }
293         }
294         set_money(dest,connected[j].money);
295         n = sendto(client.sock,"ok",2, 0, (struct sockaddr *)
            &client.cli_addr, client.clilen);

```

```

296         if (n < 0)
297         {
298             pthread_exit(0);
299         }
300     }
301 }
302
303 int check_user(char buf[]){
304     char name[bufSize+1];
305     char tmp[bufSize+1];
306     char id[10];
307     int res=-1;
308     int k;
309     FILE *file;
310     char *fname = "/home/user/project_t/us.txt";
311     file = fopen(fname,"r");
312     bzero(name,bufSize+1);
313     strcpy(name,buf);
314     if(file == NULL)
315     {
316         perror("ERROR on opening file with users");
317         exit(1);
318     }
319     int i = 0;
320     bzero(tmp,bufSize+1);
321     while(fscanf(file,"%s", tmp)!=EOF){
322         //fscanf(file,"%s", tmp);
323         if(k==0){
324             strcpy(id,tmp);
325             res=atoi(id);
326             printf("%d\n",res);
327             break;
328         }
329         if(!(i%2)){
330             k=strcmp(name,tmp);
331             printf("%s\n",tmp);
332         }
333         i++;
334     }
335     fclose(file);
336     return res;
337 }
338
339 void add(user client){
340     connected[clientsCount++]=client;
341 }
342
343 void disconnect(int sock){
344     int i = 0;

```

```

345         for (i = 0; i < clientsCount; ++i) {
346             if (connected[i].sock == sock)
347                 break;
348         }
349         if (i != clientsCount) {
350             int n = sendto(connected[i].sock, "exit", 4, 0, (
                 struct sockaddr *)&connected[i].cli_addr,
                 connected[i].clilen);
351             if (n < 0)
352             {
353                 perror("ERROR_writing_to_socket");
354                 exit(1);
355             }
356             close(sock);
357             for (++i; i < clientsCount; ++i) {
358                 connected[i - 1] = connected[i];
359             }
360             --clientsCount;
361         }
362     }
363
364 void* server_handler(void*){
365     while (1) {
366         char command[bufSize];
367         bzero(command, bufSize);
368         scanf("%s", command);
369         if (strcmp(command, "show") == 0) {
370             for (int i = 0; i < clientsCount; ++i) {
371                 printf("%d: %s\n", connected[i].sock,
                     connected[i].name);
372             }
373         } else if (strcmp(command, "disconnect") == 0) {
374             int sock = 0;
375             scanf("%d", &sock);
376             disconnect(sock);
377         } else {
378             printf("Undefined_command\n");
379         }
380     }
381 }

```

Файл server.h

```

1 #ifndef SERVER
2 #define SERVER
3 #include "user.h"
4 void *doprocessing(void *c);
5 void show_users(user client);
6 int authentication(user client);
7 int check_user(char buf[]);

```

```

8 void check_wallet(user client);
9 void transfer(user client);
10 void *server_handler(void *);
11 void disconnect(int sock);
12 void add(user client);
13 user new_connection(int sockfd);
14 void new_socket(int* sockfd, uint16_t* port);
15 #endif // SERVER

```

Файл user.cpp

```

1 #include <stdio.h>
2 #include <string.h>
3 #include <cstdlib>
4 #include "user.h"
5
6 int get_money(int usid){//узнать количество денег, имеющееся
   у пользователя
7     int i=0;
8     int res,tmp=0;
9     int k=-1;
10    char buffer[bufSize+1];
11    FILE *file;
12    char *fname="/home/user/project_t/money.txt";
13    file = fopen(fname,"r");
14    if(file == NULL)
15    {
16        perror("ERROR_on_openning_file_with_money");
17        exit(1);
18    }
19    bzero(buffer,bufSize+1);
20    while(!feof(file)){
21        fscanf(file,"%s", buffer);
22        tmp=atoi(buffer);
23        if(k==0){
24            res=tmp;
25            break;
26        }
27        if(!(i%2)){
28            if(usid==tmp)
29                k=0;
30        }
31        i++;
32    }
33    fclose(file);
34    return res;
35 }
36
37 int set_money(int uid, int value){
38     char buf[15];

```

```

39     int tmp=0;
40     int before=0;
41     int after=0;
42     int spaces=0;
43     fpos_t pos;
44     FILE *file;
45     char *fname="/home/user/project_t/money.txt";
46     file = fopen(fname,"r+");
47     if(file == NULL)
48     {
49         perror("ERROR on opening file with money");
50         exit(1);
51     }
52     int k=1;
53     int i = 0;
54     while(fscanf(file,"%s", buf)!=EOF){
55         if(k==0){
56             before=ftell(file);
57             fsetpos(file,&pos);
58             fprintf(file,"\t%i",value);
59             after=ftell(file);
60             if(before>after) {
61                 spaces += before - after;
62                 while(spaces!=0){
63                     fputc('_', file);
64                     spaces--;
65                 }
66             }
67             fflush(file);
68             break;
69         }
70         fgetpos(file, &pos);
71         if(!(i%2)){
72             tmp=atoi(buf);
73             if(tmp==uid){
74                 k=0;
75             }
76         }
77         i++;
78     }
79 }
80 }
81
82 int set_newid(){//задание нового id пользователя при регистра
    ции
83     int res=0;
84     char uid[15];
85     FILE *file;
86     char *fname = "/home/user/project_t/us.txt";

```

```

87     file = fopen(fname, "r");
88     bzero(uid, sizeof(uid));
89     if(file == NULL)
90     {
91         perror("ERROR_on_openning_file_with_users");
92         exit(1);
93     }
94     int i = 0;
95     while(fscanf(file, "%s", uid) != EOF){
96         if(i%2){
97             res=atoi(uid);
98         }
99         i++;
100     }
101     fclose(file);
102     return res+1;
103 }

```

Файл user.h

```

1  #ifndef USER
2  #define USER
3  #include <netdb.h>
4  #include <netinet/in.h>
5  #include <unistd.h>
6  #define bufSize 255
7  #define maxClients 100
8  typedef struct{
9      char name[bufSize];
10     int sock;
11     int uid;
12     int money;
13     struct sockaddr_in cli_addr;
14     socklen_t clilen;
15 }user;
16
17 //user();
18 int set_newid();
19 int get_money(int usid);
20 int set_money(int uid, int value);
21 #endif // USER

```

UDP клиент

Файл client.cpp

```

1  //client
2  #include <winsock2.h>

```

```

3 #include <ws2tcpip.h>
4 #include <stdlib.h>
5 #include <stdio.h>
6 #include <string.h>
7
8 #pragma comment (lib, "Ws2_32.lib")
9 #pragma comment (lib, "Mswsock.lib")
10 #pragma comment (lib, "AdvApi32.lib")
11
12 #define bufSize 255
13
14 typedef struct {
15     int sockfd;
16     struct sockaddr_in serv_addr;
17     int clen;
18 } Uclient;
19
20 int authentication(Uclient client);
21 void showUsers(char command[], Uclient client);
22 void checkWallet(Uclient client);
23 void transfer(Uclient client);
24 int disconnect(int sockfd, char* buf);
25
26 int main(int argc, char *argv[]) {
27     WORD wVersionRequested = MAKEWORD(2, 2);           // Stuff
28     WSADATA wsaData;
29     int sock, portno, n=0;
30     int aut;
31     struct sockaddr_in serv_addr;
32     struct hostent *server;
33     char quit[]="quit";
34     char show[]="show_users";
35     char wallet[]="check_wallet";
36     char transf[]="transfer";
37     char buffer[bufSize+1];
38     char buf[bufSize];
39
40     WSASStartup(wVersionRequested, &wsaData);
41     //portno=12345;
42     if (argc < 3) {
43         fprintf(stderr, "usage_%s_hostname_port\n", argv[0]);
44         exit(0);
45     }
46
47     portno = atoi(argv[2]);
48
49     /* Create a socket point */
50     sock = socket(AF_INET, SOCK_DGRAM, 0);

```



```

51
52 if (sock == SOCKET_ERROR) {
53     perror("ERROR opening socket");
54     exit(1);
55 }
56
57 server = gethostbyname(argv[1]);
58 if (server == NULL) {
59     fprintf(stderr, "ERROR, no such host\n");
60     exit(0);
61 }
62
63 memset((char *) &serv_addr, 0, sizeof(serv_addr));
64 serv_addr.sin_family = AF_INET;
65 serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
66 //strncpy((char *)server->h_addr, (char *)&serv_addr.
    sin_addr.s_addr, server->h_length);
67 serv_addr.sin_port = htons(portno);
68 int len = sizeof(serv_addr);
69
70
71 n = sendto(sock, "client", strlen("client"), 0, (struct
    sockaddr *) &serv_addr, len);
72 if (n < 0) {
73     perror("ERROR writing to socket");
74     closesocket(sock);
75     exit(1);
76 }
77
78 //recieve new port
79 memset(buf, 0, bufSize);
80 n = recvfrom(sock, buf, bufSize, 0, (struct sockaddr *) &
    serv_addr, &len);
81 if (n < 0) {
82     perror("ERROR reading from socket");
83     closesocket(sock);
84     exit(1);
85 }
86 disconnect(sock, buf);
87 //close old socket
88 closesocket(sock);
89 WSACleanup();
90
91 int newport = atoi(buf);
92
93 WSADATA wsa2;
94 if (WSAStartup(MAKEWORD(2, 2), &wsa2) != 0) {
95     exit(EXIT_FAILURE);
96 }

```

```

97
98 struct sockaddr_in new_addr;
99 int sockfd, new_slen = sizeof(new_addr);
100
101 //create socket
102 if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) ==
    SOCKET_ERROR) {
103     exit(EXIT_FAILURE);
104 }
105
106 //setup address structure
107 memset((char *)&new_addr, 0, new_slen);
108 new_addr.sin_family = AF_INET;
109 new_addr.sin_port = htons(newport);
110 new_addr.sin_addr.S_un.S_addr = inet_addr(argv[1]);
111
112 n = sendto(sockfd, "newclient", strlen("newclient"), 0, (
    struct sockaddr *) &new_addr, new_slen);
113 if (n < 0) {
114     perror("ERROR_reading_from_socket");
115     closesocket(sockfd);
116     exit(1);
117 }
118
119 Uclient client;
120 client.sockfd = sockfd;
121 client.serv_addr = new_addr;
122 client.clilen = sizeof(new_addr);
123
124 do {
125     aut = authentication(client); //процесс аутентифик
    ации клиента
126 } while (aut < 0);
127 while (1){
128     printf("Enter_the_command:");
129     memset(buffer, 0, bufSize+1);
130     fgets(buffer, bufSize+1, stdin);
131
132     if(strncmp(buffer, quit, sizeof(quit)-1) == 0){
133         n = sendto(client.sockfd, buffer, strlen(buffer)
            ,0, (struct sockaddr*)&client.serv_addr,
            client.clilen);
134         if (n < 0) {
135             perror("ERROR_writing_to_socket");
136             exit(1);
137         }
138         closesocket(client.sockfd);
139         break;
140     }

```

```

141     else if(strncmp(buffer,show,sizeof(show)-1) == 0){
142         showUsers(buffer, client);
143     }
144     else if(strncmp(buffer, wallet,sizeof(wallet)-1) == 0){
145         checkWallet(client);
146     }
147     else if(strncmp(buffer, transf,sizeof(transf)-1) == 0){
148         transfer(client);
149     }
150     else{
151         printf("Undefined command\n");
152     }
153 }
154 return 0;
155 }
156
157 void showUsers(char command[], Uclient client){
158     int n;
159     char buffer[bufSize+1];
160     memset(buffer, 0, bufSize);
161     strcpy(buffer,command);
162     n = sendto(client.sockfd, buffer, strlen(buffer), 0, (
163         struct sockaddr*)&client.serv_addr, client.clilen);
164     if (n < 0) {
165         perror("ERROR writing to socket");
166         closesocket(client.sockfd);
167         exit(1);
168     }
169     /* Now read server response */
170     memset(buffer, 0, bufSize);
171     n = recvfrom(client.sockfd, buffer, bufSize+1, 0, (struct
172         sockaddr*)&client.serv_addr, &client.clilen);
173     if (n < 0) {
174         perror("ERROR reading from socket");
175         closesocket(client.sockfd);
176         exit(1);
177     }
178     disconnect(client.sockfd, buffer);
179     printf("%s\n",buffer);
180 }
181 void checkWallet(Uclient client){
182     int n;
183     char buffer[bufSize+1];
184     n=sendto(client.sockfd, "wallet", 6, 0, (struct sockaddr
185         *)&client.serv_addr, client.clilen);
186     if (n < 0) {
187         perror("ERROR writing to socket");

```

```

187         closesocket(client.sockfd);
188         exit(1);
189     }
190     memset(buffer, 0, bufSize+1);
191     n = recvfrom(client.sockfd, buffer, bufSize+1, 0, (struct
        sockaddr*)&client.serv_addr, &client.clilen);
192     if (n < 0) {
193         perror("ERROR_reading_from_socket");
194         closesocket(client.sockfd);
195         exit(1);
196     }
197     disconnect(client.sockfd, buffer);
198     printf("%s\n",buffer);
199 }
200
201 void transfer(Uclient client){
202     int n;
203     char tmp[bufSize];
204     char buffer[bufSize+1];
205     memset(buffer, 0, bufSize+1);
206     n=sendto(client.sockfd, "transf", 6, 0, (struct sockaddr
        *)&client.serv_addr, client.clilen);
207     if (n < 0) {
208         perror("ERROR_writing_to_socket");
209         closesocket(client.sockfd);
210         exit(1);
211     }
212     printf("To_who_and_how_much_do_you_want_transfer_money?\n
        ");
213     scanf("%s",buffer);
214     scanf("%s",tmp);
215     strcat(buffer,"_");
216     strcat(buffer,tmp);
217     n=sendto(client.sockfd, buffer, bufSize+1, 0, (struct
        sockaddr*)&client.serv_addr, client.clilen);
218     memset(buffer, 0, bufSize+1);
219     n=recvfrom(client.sockfd, buffer, bufSize+1, 0, (struct
        sockaddr*)&client.serv_addr, &client.clilen);
220     if (n < 0) {
221         perror("ERROR_reading_from_socket");
222         closesocket(client.sockfd);
223         exit(1);
224     }
225     disconnect(client.sockfd, buffer);
226     if(strcmp(buffer,"no_match")==0){
227         printf("There_is_no_user_with_such_username\n");
228     }
229     else if(strcmp(buffer,"ok")==0){
230         printf("Operation_done\n");

```

```

231     }
232     else{
233         printf("Some_error_occurs_during_the_operation\n");
234     }
235 }
236
237 int disconnect(int sockfd, char* buf){
238     if (strcmp(buf, "exit") == 0) {
239         printf("Disconnected_from_server\n");
240         closesocket(sockfd);
241         exit(1);
242     }
243     else return -1;
244 }
245
246 int authentication(Uclient client){
247     char login[bufSize +1];
248     char reg[bufSize +1];
249     char buffer[bufSize +1];
250     char ok[]="ok";
251     char sign[]="sign_in";
252     char registration[]="register";
253     int n;
254     int res=-1;
255     memset(buffer, 0, bufSize+1);
256     printf("Sign_in_or_register\n");
257     fgets(buffer, bufSize+1, stdin);
258     //scanf("%s", &buffer);
259     if(strncmp(buffer, sign, sizeof(sign)-1) == 0){//вход существ
        теющего пользователя
260         printf("Enter_your_username:\n");
261         scanf("%s", &login);
262         n=sendto(client.sockfd, "exist", bufSize, 0, (struct
            sockaddr*)&client.serv_addr, client.clilen);//посы
            лка серверу сообщения о том, что входит существующ
            ий пользователь
263         if (n < 0) {
264             perror("ERROR_writing_to_socket");
265             closesocket(client.sockfd);
266             exit(1);
267         }
268         n=sendto(client.sockfd, login, bufSize, 0, (struct
            sockaddr*)&client.serv_addr, client.clilen);//посы
            лка серверу имени пользователя
269         if (n < 0) {
270             perror("ERROR_writing_to_socket");
271             closesocket(client.sockfd);
272             exit(1);
273         }

```

```

274     memset(buffer, 0, bufSize+1);
275     n = recvfrom(client.sockfd, buffer, bufSize+1, 0, (
        struct sockaddr*)&client.serv_addr, &client.clilen
    );//отвеем от сервера, правильны ли данные или нет
276     if (n < 0) {
277         perror("ERROR_reading_from_socket");
278         closesocket(client.sockfd);
279         exit(1);
280     }
281     disconnect(client.sockfd, buffer);
282     if(strncmp(buffer,ok,sizeof(ok)-1) == 0){//данные пра
        вильны
283         printf("Hello_%s,_you_has_successfully_logined\n"
            ,login);
284         res=1;
285     }
286     else{//данные не правильны
287         printf("No_such_username._Type_correct_username_
            or_register\n");
288         res=-1;
289     }
290 }
291 else if(strncmp(buffer,registration,sizeof(registration)
-1) == 0){//регистрация нового пользователя
292     printf("Create_new_username:\n");
293     scanf("%s", &reg);
294     n=sendto(client.sockfd,"new",bufSize, 0, (struct
        sockaddr*)&client.serv_addr, client.clilen);//посы
        лка сообщения серверу о регистрации нового пользов
        ателя
295     if (n < 0) {
296         perror("ERROR_writing_to_socket");
297         closesocket(client.sockfd);
298         exit(1);
299     }
300     n=sendto(client.sockfd,reg,bufSize, 0, (struct
        sockaddr*)&client.serv_addr, client.clilen);//посы
        лка имени нового пользователя
301     if (n < 0) {
302         perror("ERROR_writing_to_socket");
303         closesocket(client.sockfd);
304         exit(1);
305     }
306     memset(buffer, 0, bufSize+1);
307     n = recvfrom(client.sockfd, buffer, bufSize+1, 0, (
        struct sockaddr*)&client.serv_addr, &client.clilen
    );//отвеем от сервера, правильны ли данные или нет
308     if (n < 0) {
309         perror("ERROR_reading_from_socket");

```

```

310         closesocket(client.sockfd);
311         exit(1);
312     }
313     disconnect(client.sockfd, buffer);
314     if(strncmp(buffer,ok,sizeof(ok)-1) == 0){//данные пра
        вильны
315         printf("Hello\s, you has successfully registered
            and logged\n",reg);
316         res=1;
317     }
318     else{//данные не правильны
319         printf("User with this username is already
            existing. Create other username\n");
320     }
321 }
322 return res;
323 }

```