Сети ЭВМ и телекоммуникации.

Никитенко А.П.

24 декабря 2015г.

Глава 1

Задание

Разработать приложение–клиент и приложение–сервер электронного магазина. Товар в электронном магазине имеет уникальный идентификатор, именование, цену.

1.1 Функциональные требования

Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов электронного магазина
3) Поддержка одновременной работы нескольких клиентов электронного магазина через механизм нитей
4) Прием запросов на добавление или покупку товара
5) Осуществление добавления товара, учет количества единиц товара
6) Передача клиенту электронного магазина информации о товарах
7) Обработка запроса на отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Передача запросов о добавлении, покупке товаров серверу
3) Получение ответов на запросы от сервера
4) Разрыв соединения

Разработанное клиентское приложение должно предоставлять пользователю настройку IP–адреса или доменного имени сервера электронного магазина и номера порта, используемого сервером.

1.2 Нефункциональные требования

Серверное и клиентское приложения должны работать на разных ОС.

1.3 Накладываемые ограничения

Максимальное количество разных наименований, хранимых на складе и известных серверу - 1000.

Максимальная длина наименования товара - 35 символов.

Наименования товаров и пользовательские команды пишутся без разделителей.

Глава 2

Реализация по протоколу TCP

2.1 Прикладной протокол

Формат пользовательских команд - строка без разделителей.

Список команд, доступных обычному пользователю:

- buy - переход к процессу покупки товара
- showlist - просмотр информации о товарах
- exit - отключение от сервера магазина

Список команд, доступных администратору:

- create - добавить новый товар в базу
- remove - удалить товар из базы
- increase - добавить некоторое количество существующего в базе товара
- decrease - удалить некоторое количество существующего в базе товара
- showlist - просмотр информации о товарах
- exit - отключение от сервера магазина

2.2 Архитектура приложения

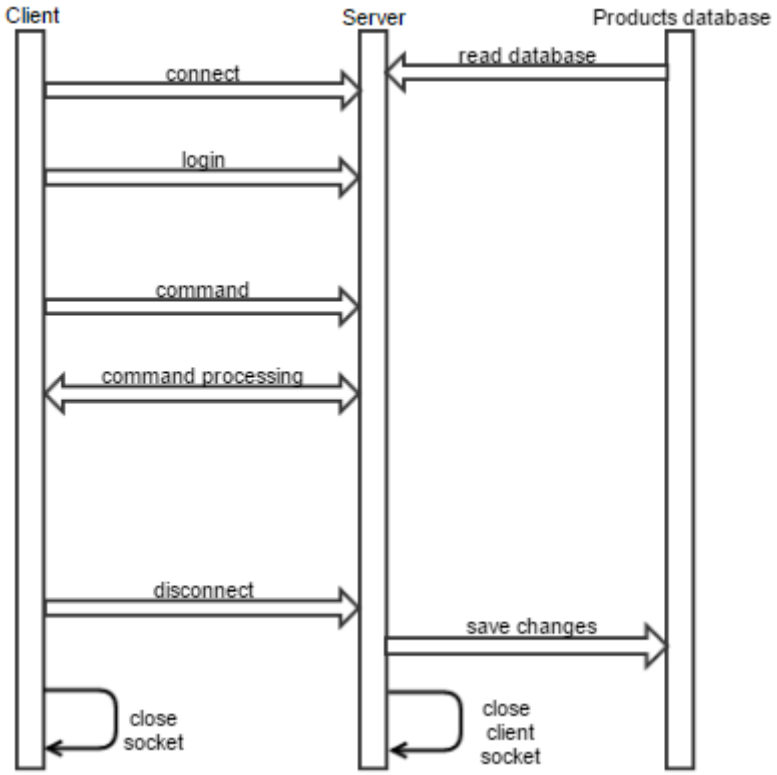


Рис.1. Диаграмма деятельности клиент-серверного приложения

При включении сервера, в программу загружаются данные о текущем состоянии магазина(список товаров, их цены, наименования и идентификаторы). Это необходимо для, того чтобы не обращаться каждый раз за данными в базу, а содержать текущие данные в программе. Кроме того, это позволяет всем клиентам всегда работать с актуальными данными. При подключении, клиент проходит процедуру идентификации. В данном проекте результат процедуры влияет только на доступные пользователю команды. Затем клиент начинает отправлять серверу доступные для него запросы, и в процесс взаимодействия сервер уточняет у пользователя необходимые для выполнения запроса данные, например, наименование покупаемого товара, цену создаваемого товара и т.д.

При отключении пользователя от сервера, сервер сохраняет информацию о состоянии магазина для последующей загрузки с новыми данными.

2.3 Тестирование

2.3.1 Описание тестового стенда и методики тестирования

Серверное приложение реализовано под ОС Linux. Тестировалось на ОС Debian 8.
Клиентское приложения реализовано под ОС Windows. Тестировалось на Windows 8.1

2.3.2 Методика тестирования

Для тестирования приложений запускается сервер электронного магазина и несколько клиентов. В процессе тестирования проверяются основные возможности приложений по передаче и приему сообщений.

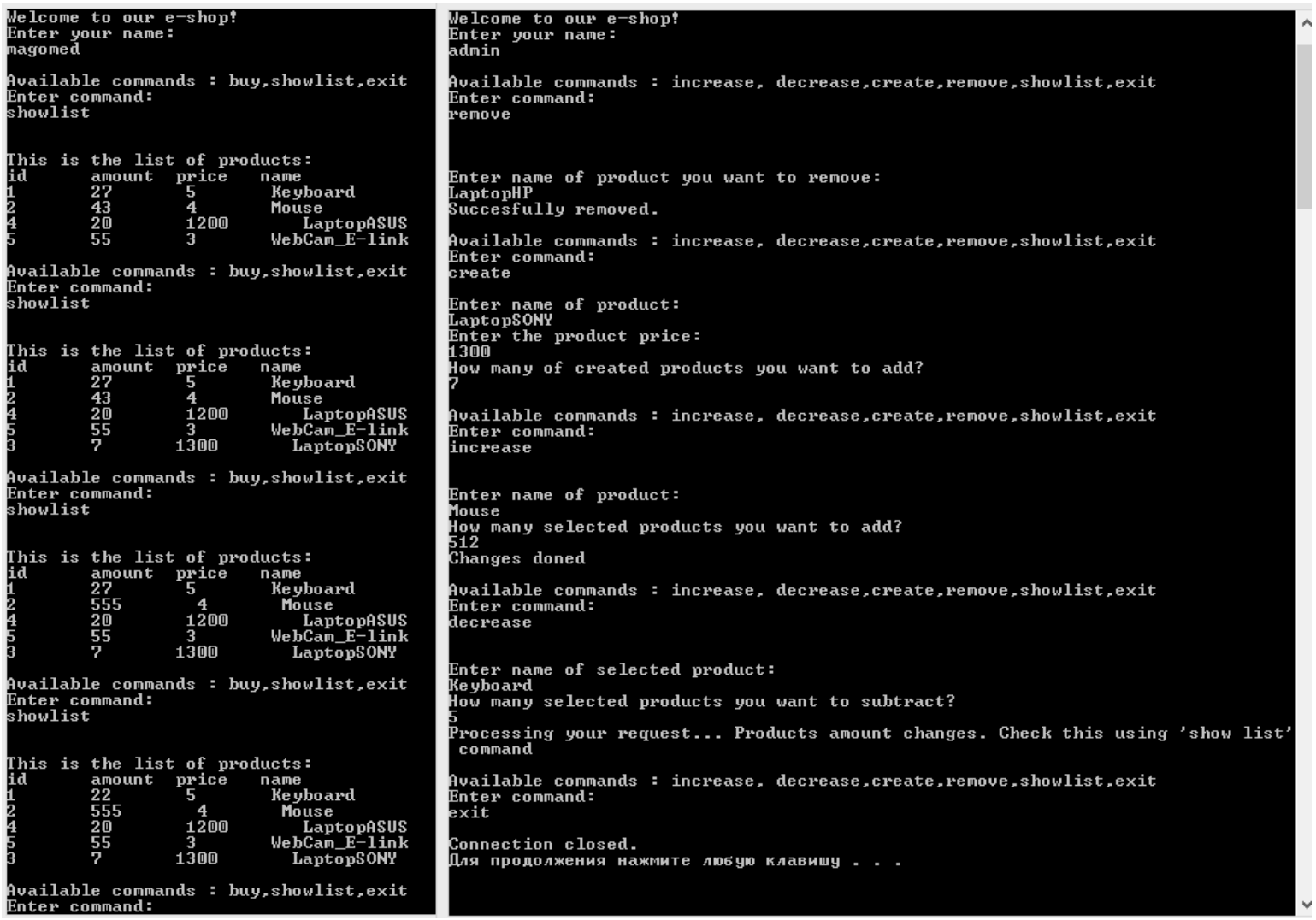
2.3.3 Ход тестирования

Для начала были проверены команды пользователя.
Подключили к серверу 2 клиентских приложения, на одном вошли как обычный пользователь, на другом - как администратор.
На стороне администратора вывели список товаров, затем на стороне обычного пользователя купили товар "LaptopASUS" в количестве 1шт. и вывели список товаров, где было зафиксировано изменение количества доступных "LaptopASUS". Затем попробовали купить 50 товаров "Mouse", и получили сообщение об ошибке, т.к. на складе нет такого количества товаров. После этого отключились от сервера.

```
Enter server's address:
192.168.0.96


Welcome to our e-shop!
Enter your name:
arsen

Available commands : buy,showlist,exit
Enter command:
buy


Enter name of selected product:
LaptopASUS
How many selected products you want to buy?
1


Processing your request... LaptopASUS

Purchase is made. Thank you for your order.

Available commands : buy,showlist,exit
Enter command:
showlist


This is the list of products:
id      amount  price   name
1       27      5       Keyboard
2       43      4       Mouse
3       15      1000       LaptopHP
4       20      1200       LaptopASUS
5       55      3       WebCam_E-link

Available commands : buy,showlist,exit
Enter command:
buy


Enter name of selected product:
Mouse
How many selected products you want to buy?
50


Processing your request... Mouse

Purchase error. Please try again.

Available commands : buy,showlist,exit
Enter command:
exit

Connection closed.
Для продолжения нажмите любую клавишу . . .
```

```
Enter server's address:
192.168.0.96


Welcome to our e-shop!
Enter your name:
admin

Available commands : increase, decrease,create,remove,showlist,exit
Enter command:
showlist


This is the list of products:
id      amount  price   name
1       27      5       Keyboard
2       43      4       Mouse
3       15      1000       LaptopHP
4       21      1200       LaptopASUS
5       55      3       WebCam_E-link

Available commands : increase, decrease,create,remove,showlist,exit
Enter command:
```

Для тестирования функций администратора, запустили 2 клиентских приложения и также вошли под разными пользователями. На стороне администратора осуществили сначала удаление из базы товара LaptopHP, затем создали новый товар LaptopSONY, увеличили количество доступных Mouse и уменьшили количество доступных Keyboard. На стороне обычного пользователя фиксировали изменения в списке товаров после каждого действия.



Также, реализация TCP-приложения была проверена на проблемы с памятью с помощью утилиты Valgrind.

Для этого собрали приложение с сохранением отладочной инфомации и запустили с помощью Valgrind:

user@debian:~/TelecomCourse/build-e-shop-Desktop_Qt_5_5_0_GCC_32bit-Debug$ valgrind --leak-check=full ./e-shop

Обнаруженные ошибки:

==1682== Invalid write of size 4
==1682== at 0x804B31E: main (main.c:90)
==1682== Address 0x436e668 is 0 bytes inside a block of size 1 alloc'd
==1682== at 0x40291CC: malloc (vg_replace_malloc.c:296)
==1682== by 0x804B311: main (main.c:89)

```
==1682== Invalid read of size 4
==1682==    at 0x804B03C: handler (main.c:19)
==1682==    by 0x41ABEFA: start_thread (pthread_create.c:309)
==1682==    by 0x42AA62D: clone (clone.S:129)
==1682==  Address 0x436e668 is 0 bytes inside a block of size 1 alloc'd
==1682==    at 0x40291CC: malloc (vg_replace_malloc.c:296)
==1682==    by 0x804B311: main (main.c:89)
```

Эти ошибки указывают на следующие строки кода:

```
main.c
19        int csock = * (int *) arg;
....
88        pthread_t thr;
89         new_sock=malloc(1);
90        *new_sock = csock;
```

Исправление:

```
89        new_sock=malloc(4);
```

```
==1682== Conditional jump or move depends on uninitialised value(s)
==1682==    at 0x8048C2B: create (admcommands.h:32)
==1682==    by 0x804AF31: admDial (dialogs.h:107)
==1682==    by 0x804B1AE: handler (main.c:45)
==1682==    by 0x41ABEFA: start_thread (pthread_create.c:309)
==1682==    by 0x42AA62D: clone (clone.S:129)
```

Ошибка исправлена освобождением памяти в функции create.

```
memset(temp1,0,sizeof(temp1));
```

Была проверена вся функциональность сервера - от момента подключения клиента его отключения, как все возможности администратора, так и возможности рядового пользователя.

Глава 3

Реализация по протоколу UDP

3.1 Прикладной протокол

Формат пользовательских команд - строка без разделителей.

Список команд, доступных обычному пользователю:

- buy - переход к процессу покупки товара
- showlist - просмотр информации о товарах
- exit - отключение от сервера магазина

Список команд, доступных администратору:

- create - добавить новый товар в базу
- remove - удалить товар из базы
- increase - добавить некоторое количество существующего в базе товара
- decrease - удалить некоторое количество существующего в базе товара
- showlist - просмотр информации о товарах
- exit - отключение от сервера магазина

3.2 Архитектура приложения

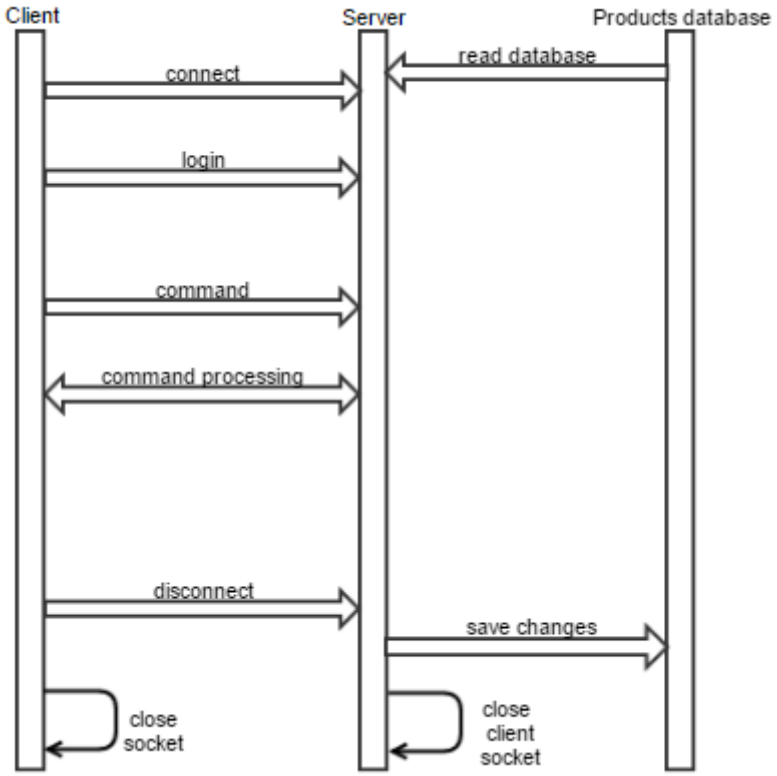


Рис.2. Диаграмма деятельности клиент-серверного приложения

При включении сервера, в программу загружаются данные о текущем состоянии магазина(список товаров, их цены, наименования и идентификаторы). Это необходимо для, того чтобы не обращаться каждый раз за данными в базу, а содержать текущие данные в программе. Кроме того, это позволяет всем клиентам всегда работать с актуальными данными. При подключении, клиент проходит процедуру идентификации. В данном проекте результат процедуры влияет только на доступные пользователю команды. Затем клиент начинает отправлять серверу доступные для него запросы, и в процесс взаимодействия сервер уточняет у пользователя необходимые для выполнения запроса данные, например, наименование покупаемого товара, цену создаваемого товара и т.д.

При отключении пользователя от сервера, сервер сохраняет информацию о состоянии магазина для последующей загрузки с новыми данными.

3.3 Тестирование

3.3.1 Описание тестового стенда и методики тестирования

Серверное приложение реализовано под ОС Windows. Тестировалось на ОС Windows 8.1
Клиентское приложения реализовано под ОС Linux. Тестировалось на Debian 8

3.3.2 Методика тестирования

Для тестирования приложений запускается сервер электронного магазина и несколько клиентов. В процессе тестирования проверяются основные возможности приложений по передаче и приему сообщений.

3.3.3 Ход тестирования

Тестирование работы приложений происходило так же, как тестирование TCP-реализации. Все возможности приложения были протестированы, результат совпали с результатами тестов TCP-реализации.

Далее UDP реализация приложения была протестирована на помехоустойчивость в сети.

Сначала были добавлены задержки 10мс, которые никак не повлияли на прием и передачу данных.

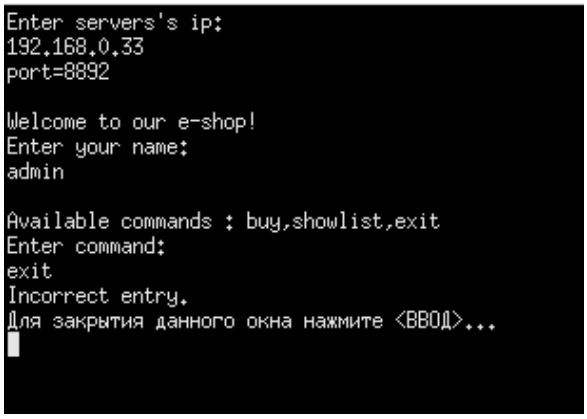
Затем были добавлены задержки 200. При такой незначительной задержке проблем не возникает.

После этого задержки были увеличены до 1000мс. Приложение все также работает корректно, все сообщение проходят, сбоев обнаружено не было, но данные приходят значительно медленнее.

Задержки - 5000мс. Приложение все еще работает, хотя и значительно медленнее. Также, иногда пакеты не доходят и приложение остается в состоянии ожидания, но все-таки, данные теряются не так часто, как ожидалось.

Затем были добавлены потери пакетов 1%.
Теперь тестировались все возможности пользователей.
При потерях =1% ошибок в работе приложения обнаружено не было.
При потерях =20% приложение перестает работать на первых секундах работы, ни разу не удалось корректно выполнить больше двух команд.

При введении дупликации 10% пакетов, работа приложения нарушилась сразу же. На рисунке ниже видно, что пользователь admin не получил доступные ему права,т.к. на сервер скорее всего вместо "admin" пришло "192.168.0.33", а также при попытке отключиться от сервера, серверу сначала пришел продублированный пакет с сообщением "admin", на что он отреагировал ошибкой, т.к. такой команды не существует, и только после этого пришла команда отсоединения от сервера и соединение было разорвано.



Затем ввели искажение 20% пакетов. Приложение почти сразу перестает работать, и зависает в ожидании каких-то данных, отказывается выполнять правильно введенные команды.

Отсюда можно сделать вывод, что приложение не готово к работе в реальных условиях, особенно если в сети резко понизится качество хотя бы одного из исследованных параметров. Данные будут теряться, искажаться и приложение зависнет спустя совсем небольшое время после начала работы.

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, реализация функционала приложения не меняется с переносом на другую ОС. В результате работы была создан электронный магазин, позволяющий вносить изменения в базу данных товаров и осуществлять покупку. Приложение работает корректно в условиях локальной сети, однако не готово к работе в реальных условиях.

4.1 Реализация для TCP

Протокол TCP удобен для реализации пользовательских приложений, так как обеспечивает установление соединения и надёжную доставку пакетов. TCP — это сложный, требующий больших затрат времени протокол, что объясняется его механизмом установления соединения, но он берет на себя заботу о гарантированной доставке пакетов, избавляя нас от необходимости включать эту функциональную возможность в прикладной протокол, что является большим плюсом. Также хочется отметить, что организовывать работу со большим количеством клиентом здесь намного легче, чем на UDP.

4.2 Реализация для UDP

В отличие от TCP, UDP — очень быстрый протокол, поскольку в нем определен самый минимальный механизм, необходимый для передачи данных. Конечно, он имеет некоторые недостатки. Сообщения поступают в любом порядке, и то, которое отправлено первым, может быть получено последним. Доставка сообщений UDP вовсе не гарантируется, сообщение может потеряться, и могут быть получены две копии одного и того же сообщения.

Для обеспечения надежной доставки сообщений приходится использоваться дополнительные инструменты. Это не целесообразно, т.к. уже есть протокол TCP, обеспечивающий надежную доставку данных.

Приложения

Описание среды разработки

Для TCP-реализации:
Серверное приложение реализовывалось в ОС Debian версии 8.1. Среда разработки — Qt Creator версии 5.5.
Клиентское приложение реализовывалось в ОС Windows 8.1. Среда разработки — Microsoft Visual Studio 2013.
Для UDP-реализации наоборот.

Листинги

TCP сервер

main.c

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include <pthread.h>
#include "product.h"
#include "dialogs.h"

#define size 1024
extern struct product goods[100];
extern int seNum = 0;
extern int cl = 0;
extern int sr = 0;

void *handler(int *arg){
    int usr = 0, n;
    int csock = *(int *)arg;
    char str[size];

    strcpy(str, "\n\nWelcome to our e-shop!\n");
    send(csock, str, strlen("\n\nWelcome to our e-shop!\n"), 0);
    memset(str, 0, sizeof(str));

    n = recv(csock, str, size, 0);
    str[n] = 0;
    memset(str, 0, sizeof(str));

    strcpy(str, "Enter your name:\n");
    send(csock, str, strlen("Enter your name:\n"), 0);
    memset(str, 0, sizeof(str));

    n = recv(csock, str, size, 0);
    str[n] = 0;

    if (!strcmp(str, "admin\r\n")){
          usr = 1;
    }
    else
          usr = 2;
```

```
    while (1){
           switch (usr) {
           case 1:
                  admDial(csock);
                  break;
           case 2:
                  usrDial(csock);
                  break;
           }
           if (cl != 0){
                  cl = 0;
                  break;
           }
    }
    memset(str, 0, sizeof(str));
    strcpy(str, "\nConnection closed.\n");
    write(csock, str, strlen("\nConnection closed.\n"));
    memset(str, 0, sizeof(str));
    close(csock);
}



int main()
{
    int lsock, csock, *new_sock;
    struct sockaddr_in serv_addr, cli_addr;
    socklen_t addr_len;
    servInit();


    lsock = socket(AF_INET, SOCK_STREAM, 0);

    bzero(&serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(1234);
    bind(lsock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));

    listen(lsock, 1024);
    addr_len = sizeof(cli_addr);


    while (csock = accept(lsock, (struct sockaddr *)&cli_addr, &addr_len))
    {

           pthread_t thr;
           new_sock = malloc(1);
           *new_sock = csock;
           if (pthread_create(&thr, NULL, handler, (void*)new_sock) < 0)
           {
                  perror("could not create thread");
                  return 1;
           }
    }
}
```

dialogs.h

```
#ifndef DIALOGS
#define DIALOGS
#include "product.h"
#include "admcommands.h"
#include "usrcommands.h"
#include "gencommands.h"
#include "server.h"
#define size 1024

struct product goods[100];
int seNum, cl;


void usrDial(int csock){
    int n, com = 0;
    char temp[size];
    strcpy(temp, "\nAvailable commands : buy,showlist,exit");
    send(csock, temp, strlen(temp), 0);
    memset(temp, 0, sizeof(temp));

    n = recv(csock, temp, size, 0);
    temp[n] = 0;
    memset(temp, 0, sizeof(temp));

    strcpy(temp, "\nEnter command:\n");
    send(csock, temp, strlen(temp), 0);
    memset(temp, 0, sizeof(temp));

    n = recv(csock, temp, size, 0);
    temp[n] = 0;
    if (!strcmp(temp, "buy\r\n"))
           com = 1;
    if (!strcmp(temp, "showlist\r\n"))
           com = 2;
    if (!strcmp(temp, "exit\r\n"))
           com = 3;
    memset(temp, 0, sizeof(temp));
    switch (com) {
    case 1:
           buy(csock);
           break;
    case 2:
           showList(csock);
           break;
    case 3:
           disc();
           break;
    default:

           strcpy(temp, "Incorrect entry.\n");
           send(csock, temp, strlen(temp), 0);
           memset(temp, 0, sizeof(temp));

           n = recv(csock, temp, size, 0);
           temp[n] = 0;
           memset(temp, 0, sizeof(temp));
           break;
    }
}
```

```
void admDial(int csock){
    int n, com = 0;
    char temp[size];
    printf("\n AdmDial \n");

    strcpy(temp, "\nAvailable commands : increase, decrease,create,remove,showlist,exit");
    send(csock, temp, strlen(temp), 0);
    memset(temp, 0, sizeof(temp));

    n = recv(csock, temp, size, 0);
    temp[n] = 0;
    memset(temp, 0, sizeof(temp));

    strcpy(temp, "\nEnter command:\n");
    send(csock, temp, strlen(temp), 0);
    memset(temp, 0, sizeof(temp));

    n = recv(csock, temp, size, 0);
    temp[n] = 0;
    if (!strcmp(temp, "increase\r\n"))
           com = 1;
    if (!strcmp(temp, "decrease\r\n"))
           com = 2;
    if (!strcmp(temp, "create\r\n"))
           com = 3;
    if (!strcmp(temp, "remove\r\n"))
           com = 4;
    if (!strcmp(temp, "showlist\r\n"))
           com = 5;
    if (!strcmp(temp, "exit\r\n"))
           com = 6;
    printf("priniato: %s", temp);
    memset(temp, 0, sizeof(temp));
    switch (com) {
    case 1:
           add(csock);
           break;
    case 2:
           decr(csock);
           break;
    case 3:
           create(csock);
           break;
    case 4:
           rem(csock);
           break;
    case 5:
           showList(csock);
           break;
    case 6:
           disc();
           break;
    default:

           strcpy(temp, "Incorrect entry.\n");
           send(csock, temp, strlen(temp), 0);
           memset(temp, 0, sizeof(temp));


           n = recv(csock, temp, size, 0);
           temp[n] = 0;
           memset(temp, 0, sizeof(temp));
                        break;
```

```
		}
	}

	#endif // DIALOGS
```

product.h

```
#ifndef PRODUCT
#define PRODUCT


struct product{
    int id;
    char name[35];
    int price;
    int amount;
};
#endif // PRODUCT
```

server.h

```
#ifndef SERVER
#define SERVER
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>


void closeServ(){
    FILE *id, *names, *prices, *amount;
    id = fopen("id.txt", "w");
    names = fopen("names.txt", "w");
    prices = fopen("prices.txt", "w");
    amount = fopen("amount.txt", "w");
    int i = 0;
    for (i = 0; i<seNum; i++) {
        if (goods[i].id != 0){
            fprintf(names, "%s", goods[i].name);
            fprintf(id, "%i\n", goods[i].id);
            fprintf(prices, "%i\n", goods[i].price);
            fprintf(amount, "%i\n", goods[i].amount);
        }
    }
    fclose(id);
    fclose(names);
    fclose(prices);
    fclose(amount);
}


void servInit(){
    FILE *id, *names, *prices, *amount;
    id = fopen("id.txt", "r");
    names = fopen("names.txt", "r");
```

```
    prices = fopen("prices.txt", "r");
    amount = fopen("amount.txt", "r");
    int i = 0;
    while (!feof(names)) {
           fgets(goods[i].name, 35, names);
           fscanf(id, "%i", &goods[i].id);
           fscanf(prices, "%i", &goods[i].price);
           fscanf(amount, "%i", &goods[i].amount);
           i++;
           seNum++;
    }
    fclose(id);
    fclose(names);
    fclose(prices);
    fclose(amount);
}

#endif // SERVER
```

gencommands.h

```
#ifndef GENCOMMANDS
#define GENCOMMANDS
#include "product.h"
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include "server.h"
#define size 1024

struct product goods[100];
int seNum, cl;


void showList(int csock){
    char temp[size];
    int i, n;
    printf("\n show list \n");
    sleep(1);

    strcpy(temp, "\n\nThis is the list of products:\n");
    send(csock, temp, strlen("\n\nThis is the list of products:\n"), 0);
    memset(temp, 0, sizeof(temp));
    sleep(1);

    n = recv(csock, temp, size, 0);
    temp[n] = 0;
    memset(temp, 0, sizeof(temp));

    strcpy(temp, "id     amount price  name\n");
    send(csock, temp, strlen("id     amount     price     name\n"), 0);
    memset(temp, 0, sizeof(temp));
    sleep(1);


    n = recv(csock, temp, size, 0);
    temp[n] = 0;
    memset(temp, 0, sizeof(temp));
```

```
    sprintf(temp, "%i", seNum);
    send(csock, temp, strlen(temp), 0);
    memset(temp, 0, sizeof(temp));
    sleep(1);

    n = recv(csock, temp, size, 0);
    temp[n] = 0;
    memset(temp, 0, sizeof(temp));

    for (i = 0; i<seNum - 1; i++){
          if (goods[i].id != 0){
                sprintf(temp, "%i       %i       %i       %s", goods[i].id, goods[i].amount,
goods[i].price, goods[i].name);
                send(csock, temp, strlen(temp), 0);
                memset(temp, 0, sizeof(temp));

                n = recv(csock, temp, size, 0);
                temp[n] = 0;
                memset(temp, 0, sizeof(temp));

          }
    }
    n = recv(csock, temp, size, 0);
    temp[n] = 0;
    memset(temp, 0, sizeof(temp));
}


void disc(){
    closeServ();
    cl = 1;
    //close(csock);
}
#endif // GENCOMMANDS
```

admcommands.h

```
#ifndef ADMCOMMANDS
#define ADMCOMMANDS
#include "product.h"
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#define size 1024

struct product goods[100];
int seNum, cl, sr;


void create(int csock){
    char temp[size];
    char temp1[size];
    char str[size];
    int n, am, pr, i = 0, j = 0;
    printf("\n create \n");
```

```
    strcpy(str, "\nEnter name of product:\n");
    write(csock, str, strlen("\nEnter name of product:\n"));
    memset(str, 0, sizeof(str));


    n = recv(csock, temp, size, 0);
    temp[n] = 0;


    strncpy(temp1, temp, strlen(temp) - 2);
    memset(temp, 0, sizeof(temp));
    strcat(temp1, "\n");
    strcpy(goods[seNum - 1].name, temp1);
    memset(temp1, 0, sizeof(temp1));

    strcpy(str, "Enter the product price: \n");
    write(csock, str, strlen(str));
    memset(str, 0, sizeof(str));


    n = recv(csock, str, 1024, 0);
    str[n] = 0;
    pr = atoi(str);
    memset(str, 0, sizeof(str));
    goods[seNum - 1].price = pr;

    strcpy(str, "How many of created products you want to add? \n");
    write(csock, str, strlen(str));
    memset(str, 0, sizeof(str));


    n = recv(csock, str, 1024, 0);
    str[n] = 0;
    am = atoi(str);
    memset(str, 0, sizeof(str));
    goods[seNum - 1].amount = am;
    goods[seNum - 1].id = 1;

    for (j = 0; j<seNum - 1; j++)
    for (i = 0; i<seNum - 1; i++)
    {
           if (goods[seNum - 1].id == goods[i].id){
                  goods[seNum - 1].id++;
           }
    }
    seNum++;

    n = recv(csock, str, size, 0);
    str[n] = 0;
    memset(str, 0, sizeof(str));
}



void rem(int csock){
    char temp[size];
    char str[size];
    int n, i, flag = 0, r_num = 0;
    printf("\n remove \n");
    strcpy(str, "\n\n\nEnter name of product you want to remove:\n");
    write(csock, str, strlen(str));
    memset(str, 0, sizeof(str));

    n = recv(csock, temp, size, 0);
```

```
    temp[n] = 0;


    for (i = 0; i<seNum - 1; i++){
           if (!strncmp(goods[i].name, temp, strlen(temp) - 2)){
                  r_num = i;
                  flag++;
           }
    }

    if (flag>0){
           for (i = r_num; i<seNum - 1; i++){
                  goods[i].amount = goods[i + 1].amount;
                  goods[i].id = goods[i + 1].id;
                  goods[i].price = goods[i + 1].price;
                  strcpy(goods[i].name, goods[i + 1].name);

           }
           bzero(&goods[seNum - 1], sizeof(struct product));
           seNum--;

           strcpy(str, "Succesfully removed.\n");
           write(csock, str, strlen(str));
           memset(str, 0, sizeof(str));
    }
    else {
           strcpy(str, "No such product. Try again.\n");
           write(csock, str, strlen(str));
           memset(str, 0, sizeof(str));
    }
    memset(temp, 0, sizeof(temp));

    n = recv(csock, str, size, 0);
    str[n] = 0;
    memset(str, 0, sizeof(str));

    n = recv(csock, str, size, 0);
    str[n] = 0;
    memset(str, 0, sizeof(str));
}




void add(int csock){
    char temp[size];
    char str[size];
    int n, am, i, flag = 0;

    strcpy(str, "\n\nEnter name of product:\n");
    write(csock, str, strlen(str));
    memset(str, 0, sizeof(str)); \


           n = recv(csock, temp, size, 0);
    temp[n] = 0;


    strcpy(str, "How many selected products you want to add?\n");
    write(csock, str, strlen(str));
    memset(str, 0, sizeof(str)); \


           n = recv(csock, str, size, 0);
```

```
    str[n] = 0;
    am = atoi(str);
    memset(str, 0, sizeof(str));
    printf("am=%i", am);


    for (i = 0; i<seNum - 1; i++){
           if (!strncmp(goods[i].name, temp, strlen(temp) - 2)){
                  goods[i].amount = goods[i].amount + am;
                  flag++;
           }
    }
    if (flag == 1){
           strcpy(str, "Changes doned\n");
           write(csock, str, strlen(str));
           memset(str, 0, sizeof(str));
    }
    else {
           strcpy(str, "No such product. Try again.\n");
           write(csock, str, strlen(str));
           memset(str, 0, sizeof(str));
    }
    memset(temp, 0, sizeof(temp));

    n = recv(csock, str, size, 0);
    str[n] = 0;
    memset(str, 0, sizeof(str));

    n = recv(csock, str, size, 0);
    str[n] = 0;
    memset(str, 0, sizeof(str));
}



void decr(int csock){
    char temp[size];
    char str[size];
    int n, am, i, flag = 0;

    printf("\n decrease \n");

    strcpy(str, "\n\nEnter name of selected product:\n");
    write(csock, str, strlen(str));
    memset(str, 0, sizeof(str));

    n = recv(csock, temp, 1024, 0);
    temp[n] = 0;/*
                        strcpy(str,"You selected: ");
                        strcat(str,temp);
                        write(csock,str,strlen(str));*/

    strcpy(str, "How many selected products you want to subtract?\n");
    write(csock, str, strlen(str));
    memset(str, 0, sizeof(str));

    n = recv(csock, str, size, 0);
    str[n] = 0;
    am = atoi(str);
    memset(str, 0, sizeof(str));
    printf("am=%i\n", am);


    strcpy(str, "Processing your request... ");
    //strcat(str,temp);
```

```
    write(csock, str, strlen(str));
    memset(str, 0, sizeof(str));

    /*sprintf(str,"%s %i\n","Amount: ",am);
    write(csock,str,strlen(str));
    memset(str,0,sizeof(str));*/
    n = recv(csock, str, size, 0);
    str[n] = 0;
    memset(str, 0, sizeof(str));

    for (i = 0; i<seNum - 1; i++){
           if (!strncmp(goods[i].name, temp, strlen(temp) - 2)){
                  if (am>goods[i].amount)
                  {
                         flag = 0;
                  }
                  else
                  {
                         goods[i].amount = goods[i].amount - am;
                         flag++;
                  }
           }
    }

    if (flag == 1){
           strcpy(str, "Products amount changes. Check this using 'show list' command\n");
           write(csock, str, strlen(str));
           memset(str, 0, sizeof(str));
    }
    else {
           strcpy(str, "Error(no such product or too few products). Please try again.\n");
           write(csock, str, strlen(str));
           memset(str, 0, sizeof(str));
    }
    memset(temp, 0, sizeof(temp));

    n = recv(csock, str, size, 0);
    str[n] = 0;
    memset(str, 0, sizeof(str));



    n = recv(csock, str, size, 0);
    str[n] = 0;
    memset(str, 0, sizeof(str));
}

#endif // ADMCOMMANDS
```

usrcommands.h

```
#ifndef USRCOMMANDS
#define USRCOMMANDS
#include "product.h"
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
```

```
#include <string.h>
#define size 1024

struct product goods[100];
int seNum, cl;


void buy(int csock){
    char temp[size];
    char str[size];
    int n, am, i, flag = 0;
    strcpy(str, "\n\nEnter name of selected product:\n");
    write(csock, str, strlen(str));
    memset(str, 0, sizeof(str));

    n = recv(csock, temp, size, 0);
    temp[n] = 0;

    strcpy(str, "How many selected products you want to buy?\n");
    write(csock, str, strlen(str));
    memset(str, 0, sizeof(str));

    n = recv(csock, str, size, 0);
    str[n] = 0;
    am = atoi(str);
    memset(str, 0, sizeof(str));


    sprintf(str, "\n\n%s %s\n", "Processing your request...", temp);
    write(csock, str, strlen(str));
    memset(str, 0, sizeof(str));
    /*sprintf(str,"%s %i\n","Amount: ",am);
    write(csock,str,strlen(str));
    memset(str,0,sizeof(str));*/

    n = recv(csock, str, size, 0);
    str[n] = 0;
    memset(str, 0, sizeof(str));

    for (i = 0; i<seNum - 1; i++){
           if (!strncmp(goods[i].name, temp, strlen(temp) - 2)){
                  if (am>goods[i].amount)
                  {
                         flag = 0;
                  }
                  else
                  {
                         goods[i].amount = goods[i].amount - am;
                         flag++;
                  }
           }
    }
    if (flag == 1){
           strcpy(str, "Purchase is made. Thank you for your order.\n");
           write(csock, str, strlen(str));
           memset(str, 0, sizeof(str));
    }
    else {
           strcpy(str, "Purchase error. Please try again.\n");
           write(csock, str, strlen(str));
           memset(str, 0, sizeof(str));
    }
    memset(temp, 0, sizeof(temp));

    n = recv(csock, str, size, 0);
```

```
    str[n] = 0;
    memset(str, 0, sizeof(str));



    n = recv(csock, str, size, 0);
    str[n] = 0;
    memset(str, 0, sizeof(str));

}

#endif // USRCOMMANDS
```

TCP клиент

main.c

```
#define WIN32_LEAN_AND_MEAN
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <winsock2.h>
#include <windows.h>
#include <locale.h>
#include <malloc.h>
#include <Ws2tcpip.h>
#pragma comment(lib, "Ws2_32.lib")
#include "dialogs.h"

#define PORT 1234
//#define SERVERADDR "10.1.99.25"
#define size  1024

int cl = 0;


int main()
{
    setlocale(LC_ALL, "Russian");
    char buffer[size];
    int i, n, usr = 0;
    char str[size];
    char inp[size];
    char SERVERADDR[size];
    printf("Enter server's address:\n");
    scanf("%s",SERVERADDR);
    if (WSAStartup(0x202, (WSADATA *)&buffer[0]))
    {
           printf("WSAStart error %d\n", WSAGetLastError());
           _getch();
           return -1;
    }

    SOCKET sock;
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock < 0)
```

```
{
	printf("Socket() error %d\n", WSAGetLastError());
	_getch();
	return -1;
}
sockaddr_in dest_addr;
dest_addr.sin_family = AF_INET;
dest_addr.sin_port = htons(PORT);
HOSTENT *hst;

if (inet_addr(SERVERADDR) != INADDR_NONE)
	dest_addr.sin_addr.s_addr = inet_addr(SERVERADDR);
else
if (hst = gethostbyname(SERVERADDR))
	((unsigned long *)&dest_addr.sin_addr)[0] =
	((unsigned long **)hst->h_addr_list)[0][0];
else
{
	printf("Invalid address %s\n", SERVERADDR);
	closesocket(sock);
	WSACleanup();
	_getch();
	return -1;
}

if (connect(sock, (sockaddr *)&dest_addr,
	sizeof(dest_addr)))
{
	printf("Connect error %d\n", WSAGetLastError());
	_getch();
	return -1;
}
else
{
	n = recv(sock, str, size, 0);
	str[n] = 0;
	printf(str);
	memset(str, 0, sizeof(str));

	strcpy(str, "1");
	send(sock, str, strlen("1"), 0);
	memset(str, 0, sizeof(str));

	n = recv(sock, str, size, 0);
	str[n] = 0;
	printf(str);
	memset(str, 0, sizeof(str));

	scanf("%s", inp);
	if (!strcmp(inp, "admin")){
		usr = 1;
	}
	else
		usr = 0;
	while (cl == 0){
		switch (usr) {
		case 1:
			memset(inp, 0, sizeof(inp));
			send(sock, "admin\r\n", strlen("admin\r\n"), 0);
			admDial(sock);
			break;
		case 0:
			memset(inp, 0, sizeof(inp));
			send(sock, "user\r\n", strlen("user\r\n"), 0);
			usrDial(sock);
```

```
                        break;
                }
            }
            n = recv(sock, str, size, 0);
            str[n] = 0;
            printf(str);
            memset(str, 0, sizeof(str));
            closesocket(sock);
            WSACleanup();
            return 0;
    }
}
```

dialogs.h

```
#ifndef DIALOGS
#define DIALOGS
#define WIN32_LEAN_AND_MEAN
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <winsock2.h>
#include <windows.h>
#include <locale.h>
#include <malloc.h>
#include <Ws2tcpip.h>
#pragma comment(lib, "Ws2_32.lib")
/*#include "product.h"*/
#include "admcommands.h"
/*#include "usrcommands.h"
#include "gencommands.h"
#include "server.h"*/
#include "gencommands.h"
#define size 1024

extern int cl;

void usrDial(int sock){
    char str[size];
    char temp[1024];
    int com = 0;
    int n;

    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    strcpy(temp, "\nAvailable commands : showlist,buy,exit");
    send(sock, temp, strlen(temp), 0);
    memset(temp, 0, sizeof(temp));

    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
```

```
    strcat(str, "\r\n");
    if (!strcmp(str, "showlist\r\n")){
           com = 1;
           send(sock, "showlist\r\n", strlen("showlist\r\n"), 0);
           showList(sock);
    }
    if (!strcmp(str, "buy\r\n")){
           com = 2;
           send(sock, "buy\r\n", strlen("buy\r\n"), 0);
           decrease(sock);
    }
    if (!strcmp(str, "exit\r\n")){
           com = 3;
           send(sock, "exit\r\n", strlen("exit\r\n"), 0);
           cl = 1;
    //     showList(sock);
    }
    if (com == 0){

           send(sock, "incorrect\r\n", strlen("incorrect\r\n"), 0);

           n = recv(sock, str, size, 0);
           str[n] = 0;
           printf(str);
           memset(str, 0, sizeof(str));
    }
    memset(str, 0, sizeof(str));
}




void admDial(int sock){
    char str[size];
    char temp[1024];
    int com = 0;
    int n;

    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    strcpy(temp, "\nAvailable commands : increase, decrease,create,remove,showlist,exit");
    send(sock, temp, strlen(temp), 0);
    memset(temp, 0, sizeof(temp));

    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");

    if (!strcmp(str, "increase\r\n"))
    {
           com = 1;
           send(sock, "increase\r\n", strlen("increase\r\n"), 0);
           increase(sock);
    }
    if (!strcmp(str, "decrease\r\n")){
           com = 2;
```

```
            send(sock, "decrease\r\n", strlen("decrease\r\n"), 0);
            decrease(sock);
    }
    if (!strcmp(str, "create\r\n"))
    {
            com = 3;
            send(sock, "create\r\n", strlen("create\r\n"), 0);
            create(sock);
    }
    if (!strcmp(str, "remove\r\n")){
            com = 4;
            send(sock, "remove\r\n", strlen("remove\r\n"), 0);
            rem(sock);
    }
    if (!strcmp(str, "showlist\r\n")){
            com = 5;
            send(sock, "showlist\r\n", strlen("showlist\r\n"), 0);
            showList(sock);
    }
    if (!strcmp(str, "exit\r\n")){
            com = 6;
            send(sock, "exit\r\n", strlen("exit\r\n"), 0);
            //closesocket(sock);
            cl = 1;
            //showList(sock);
    }


    if (com == 0){

            send(sock, "incorrect\r\n", strlen("incorrect\r\n"), 0);

            n = recv(sock, str, size, 0);
            str[n] = 0;
            printf(str);
            memset(str, 0, sizeof(str));
    }
    memset(str, 0, sizeof(str));
}


#endif // DIALOGS
```

admcommands.h

```
#ifndef ADMCOMMANDS
#define ADMCOMMANDS
#define WIN32_LEAN_AND_MEAN
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <winsock2.h>
#include <windows.h>
#include <locale.h>
#include <malloc.h>
#include <Ws2tcpip.h>
#pragma comment(lib, "Ws2_32.lib")
#define size 1024

void decrease(int sock){
    char str[size];
```

```
    char temp[1024];
    int counter, i;
    int n;

    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    send(sock, str, strlen(str), 0);
    memset(temp, 0, sizeof(temp));

    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    send(sock, str, strlen(str), 0);
    memset(temp, 0, sizeof(temp));




    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));


    strcpy(temp, "1");
    send(sock, temp, strlen("1"), 0);
    memset(temp, 0, sizeof(temp));


    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    strcpy(temp, "1");
    send(sock, temp, strlen("1"), 0);
    memset(temp, 0, sizeof(temp));
}


void increase(int sock){
    char str[size];
    char temp[1024];
    int counter, i;
    int n;


    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    send(sock, str, strlen(str), 0);
    memset(temp, 0, sizeof(temp));
```

```
    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    send(sock, str, strlen(str), 0);
    memset(temp, 0, sizeof(temp));




    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));



    strcpy(temp, "1");
    send(sock, temp, strlen("1"), 0);
    memset(temp, 0, sizeof(temp));
}


void rem(int sock){
    char str[size];
    char temp[1024];
    int counter, i;
    int n;


    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    send(sock, str, strlen(str), 0);
    memset(temp, 0, sizeof(temp));

    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));



    strcpy(temp, "1");
    send(sock, temp, strlen("1"), 0);
    memset(temp, 0, sizeof(temp));



    strcpy(temp, "1");
    send(sock, temp, strlen("1"), 0);
    memset(temp, 0, sizeof(temp));
}
```

```
void create(int sock){
    char str[size];
    char temp[1024];
    int counter, i;
    int n;


    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    send(sock, str, strlen(str), 0);
    memset(temp, 0, sizeof(temp));



    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    send(sock, str, strlen(str), 0);
    memset(temp, 0, sizeof(temp));



    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    send(sock, str, strlen(str), 0);
    memset(temp, 0, sizeof(temp));


    strcpy(temp, "1");
    send(sock, temp, strlen("1"), 0);
    memset(temp, 0, sizeof(temp));
}
#endif // ADMCOMMANDS
```

gencommands.h

```
#ifndef GENCOMMANDS
#define GENCOMMANDS
#define WIN32_LEAN_AND_MEAN
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <winsock2.h>
#include <windows.h>
#include <locale.h>
#include <malloc.h>
#include <Ws2tcpip.h>
#pragma comment(lib, "Ws2_32.lib")
#define size 1024

void showList(int sock){
    char str[size];
    char temp[1024];
    int counter,i;
    int n;

    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));


    strcpy(temp, "1");
    send(sock, temp, strlen("1"), 0);
    memset(temp, 0, sizeof(temp));

    n = recv(sock, str, size, 0);
    str[n] = 0;
    printf(str);
    memset(str, 0, sizeof(str));

    strcpy(temp, "1");
    send(sock, temp, strlen("1"), 0);
    memset(temp, 0, sizeof(temp));

    n = recv(sock, str, size, 0);
    str[n] = 0;
    counter = atoi(str);
    memset(str, 0, sizeof(str));

    strcpy(temp, "1");
    send(sock, temp, strlen("1"), 0);
    memset(temp, 0, sizeof(temp));

    for (i = 0; i < counter - 1; i++){
           n = recv(sock, str, size, 0);
           str[n] = 0;
           printf(str);
           memset(str, 0, sizeof(str));


           strcpy(temp, "1");
           send(sock, temp, strlen("1"), 0);
           memset(temp, 0, sizeof(temp));
    }

}
#endif // GENCOMMANDS
```

UDP сервер

```
#include<stdio.h>
#include<winsock2.h>
#include<process.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>

#pragma comment(lib,"ws2_32.lib") //Winsock Library
DWORD WINAPI threadHandler(LPVOID);
#define size 1024
#define BUFLEN 1024 //Max length of buffer
#define PORT 8888   //The port on which to listen for incoming data

struct product{
    int id;
    char name[35];
    int price;
    int amount;
};
typedef struct {
    int sockfd;
    struct sockaddr_in client;
    int clilen;
} btClient;

int cl = 0, clientsNum = 0, seNum = 0;
struct product goods[100];

void closeServ(){
    FILE *id, *names, *prices, *amount;
    id = fopen("id.txt", "w");
    names = fopen("names.txt", "w");
    prices = fopen("prices.txt", "w");
    amount = fopen("amount.txt", "w");
    int i = 0;
    for (i = 0; i<seNum; i++) {
          if (goods[i].id != 0){
                fprintf(names, "%s", goods[i].name);
                fprintf(id, "%i\n", goods[i].id);
                fprintf(prices, "%i\n", goods[i].price);
                fprintf(amount, "%i\n", goods[i].amount);
          }
    }
    fclose(id);
    fclose(names);
    fclose(prices);
    fclose(amount);
}

void disc(){
    closeServ();
    cl = 1;
}

void create(btClient client){
    char temp[size];
```

```
    char temp1[size];
    char str[size];
    int n, am, pr, i = 0, j = 0;
    int csock = client.sockfd;
    struct sockaddr_in si_other = client.client;
    int slen = client.clilen;

    seNum++;
    sendto(csock, "\nEnter name of product:\n", strlen("\nEnter name of product:\n"), 0, (struct
sockaddr *)&si_other, slen);

    n = recvfrom(csock, temp, size, 0, (struct sockaddr *)&si_other, &slen);
    temp[n] = 0;
    printf("\n\n%s\n\n", temp);

    strcpy(goods[seNum - 1].name, temp);
    memset(temp, 0, sizeof(temp));

    sendto(csock, "Enter the product price: \n", strlen("Enter the product price: \n"), 0, (struct
sockaddr *)&si_other, slen);

    n = recvfrom(csock, str, size, 0, (struct sockaddr *)&si_other, &slen);
    str[n] = 0;
    pr = atoi(str);
    memset(str, 0, sizeof(str));
    goods[seNum - 1].price = pr;

    sendto(csock, "How many of created products you want to add? \n", strlen("How many of created
products you want to add? \n"), 0, (struct sockaddr *)&si_other, slen);

    n = recvfrom(csock, str, size, 0, (struct sockaddr *)&si_other, &slen);
    str[n] = 0;
    am = atoi(str);
    memset(str, 0, sizeof(str));

    goods[seNum - 1].amount = am;
    goods[seNum - 1].id = 1;

    for (j = 0; j<seNum - 1; j++)
    for (i = 0; i<seNum - 1; i++)
    {
           if (goods[seNum - 1].id == goods[i].id){
                  goods[seNum - 1].id++;
           }
    }
}

void rem(btClient client){
    char temp[size];
    char str[size];
    int n, i = 0, flag = 0, r_num = 0;
    int csock = client.sockfd;
    struct sockaddr_in si_other = client.client;
    int slen = client.clilen;
    sendto(csock, "\n\n\nEnter name of product you want to remove:\n", strlen("\n\n\nEnter name of
product you want to remove:\n"), 0, (struct sockaddr *)&si_other, slen);

    n = recvfrom(csock, temp, size, 0, (struct sockaddr *)&si_other, &slen);
    temp[n] = 0;


    for (i = 0; i<seNum; i++){
           if (!strncmp(goods[i].name, temp, strlen(temp) - 2)){
                  r_num = i;
                  flag++;
```

```
		}
	}

	if (flag>0){
		for (i = r_num; i<seNum - 1; i++){
			goods[i].amount = goods[i + 1].amount;
			goods[i].id = goods[i + 1].id;
			goods[i].price = goods[i + 1].price;
			strcpy(goods[i].name, goods[i + 1].name);

		}
		memset(&goods[seNum - 1], 0, sizeof(struct product));
		seNum--;

		sendto(csock, "Succesfully removed.\n", strlen("Succesfully removed.\n"), 0, (struct
sockaddr *)&si_other, slen);
	}
	else {
		sendto(csock, "No such product. Try again.\n", strlen("No such product. Try again.\n"),
0, (struct sockaddr *)&si_other, slen);
	}
	memset(temp, 0, sizeof(temp));
}

void add(btClient client){
	char temp[size];
	char str[size];
	int n, am, i, flag = 0;
	int csock = client.sockfd;
	struct sockaddr_in si_other = client.client;
	int slen = client.clilen;

	sendto(csock, "\n\nEnter name of product:\n", strlen("\n\nEnter name of product:\n"), 0, (struct
sockaddr *)&si_other, slen);

	n = recvfrom(csock, temp, size, 0, (struct sockaddr *)&si_other, &slen);
	temp[n] = 0;

	sendto(csock, "How many selected products you want to add?\n", strlen("How many selected
products you want to add?\n"), 0, (struct sockaddr *)&si_other, slen);

	n = recvfrom(csock, str, size, 0, (struct sockaddr *)&si_other, &slen);
	str[n] = 0;
	am = atoi(str);
	memset(str, 0, sizeof(str));
	printf("am=%i", am);


	for (i = 0; i<seNum; i++){
		if (!strncmp(goods[i].name, temp, strlen(temp) - 2)){
			goods[i].amount = goods[i].amount + am;
			flag++;
		}
	}
	if (flag == 1){
		sendto(csock, "Changes doned\n", strlen("Changes doned\n"), 0, (struct sockaddr
*)&si_other, slen);

	}
	else {
		sendto(csock, "No such product. Try again.\n", strlen("No such product. Try again.\n"),
0, (struct sockaddr *)&si_other, slen);
	}
	memset(temp, 0, sizeof(temp));
}
```

```
void decr(btClient client){
    char temp[size];
    char str[size];
    int n, am, i, flag = 0;
    int csock = client.sockfd;
    struct sockaddr_in si_other = client.client;
    int slen = client.clilen;

    sendto(csock, "\n\nEnter name of selected product:\n", strlen("\n\nEnter name of selected
product:\n"), 0, (struct sockaddr *)&si_other, slen);

    n = recvfrom(csock, temp, size, 0, (struct sockaddr *)&si_other, &slen);
    temp[n] = 0;

    sendto(csock, "How many selected products you want to subtract?\n", strlen("How many selected
products you want to subtract?\n"), 0, (struct sockaddr *)&si_other, slen);

    n = recvfrom(csock, str, size, 0, (struct sockaddr *)&si_other, &slen);
    str[n] = 0;
    am = atoi(str);
    memset(str, 0, sizeof(str));
    printf("am=%i\n", am);

    sendto(csock, "Processing your request... \n", strlen("Processing your request... \n"), 0,
(struct sockaddr *)&si_other, slen);


    for (i = 0; i<seNum; i++){
        if (!strncmp(goods[i].name, temp, strlen(temp) - 2)){
            if (am>goods[i].amount)
            {
                flag = 0;
            }
            else
            {
                goods[i].amount = goods[i].amount - am;
                flag++;
            }
        }
    }
    if (flag == 1){
        sendto(csock, "Products amount changes. Check this using 'show list' command\n",
strlen("Products amount changes. Check this using 'show list' command\n"), 0, (struct sockaddr
*)&si_other, slen);
    }
    else {
        sendto(csock, "Error(no such product or too few products). Please try again.\n",
strlen("Error(no such product or too few products). Please try again.\n"), 0, (struct sockaddr
*)&si_other, slen);
    }
    memset(temp, 0, sizeof(temp));
}

void buy(btClient client){
    char temp[size];
    char str[size];
    int n, am, i, flag = 0;
    int csock = client.sockfd;
    struct sockaddr_in si_other = client.client;
    int slen = client.clilen;
    sendto(csock, "\n\nEnter name of selected product:\n", strlen("\n\nEnter name of selected
product:\n"), 0, (struct sockaddr *)&si_other, slen);

    n = recvfrom(csock, temp, size, 0, (struct sockaddr *)&si_other, &slen);
```

```
    temp[n] = 0;

    sendto(csock, "How many selected products you want to buy?\n", strlen("How many selected
products you want to buy?\n"), 0, (struct sockaddr *)&si_other, slen);

    n = recvfrom(csock, str, size, 0, (struct sockaddr *)&si_other, &slen);
    str[n] = 0;
    am = atoi(str);
    memset(str, 0, sizeof(str));


    sprintf(str, "\n\n%s %s\n", "Processing your request...", temp);
    sendto(csock, str, strlen(str), 0, (struct sockaddr *)&si_other, slen);
    memset(str, 0, sizeof(str));

    for (i = 0; i<seNum - 1; i++){
          if (!strncmp(goods[i].name, temp, strlen(temp) - 2)){
                if (am>goods[i].amount)
                {
                      flag = 0;
                }
                else
                {
                      goods[i].amount = goods[i].amount - am;
                      flag++;
                }
          }
    }
    if (flag == 1){
          sendto(csock, "Purchase is made. Thank you for your order.\n", strlen("Purchase is made.
Thank you for your order.\n"), 0, (struct sockaddr *)&si_other, slen);
    }
    else {
          sendto(csock, "Purchase error. Please try again.\n", strlen("Purchase error. Please try
again.\n"), 0, (struct sockaddr *)&si_other, slen);
    }
    memset(temp, 0, sizeof(temp));
}

void showList(btClient client){
    char temp[size];
    int i, n;
    int csock = client.sockfd;
    struct sockaddr_in si_other = client.client;
    int slen = client.clilen;
    Sleep(5);
    n = recvfrom(csock, temp, size, 0, (struct sockaddr *) &si_other, &slen);
    memset(temp, 0, sizeof(temp));
    Sleep(5);

    strcpy(temp, "\n\nThis is the list of products:\n");
    sendto(csock, temp, strlen(temp), 0, (struct sockaddr *)&si_other, slen);
    memset(temp, 0, sizeof(temp));
    Sleep(5);

    strcpy(temp, "id     amount price  name\n");
    sendto(csock, temp, strlen(temp), 0, (struct sockaddr *)&si_other, slen);
    memset(temp, 0, sizeof(temp));
    Sleep(5);


    sprintf(temp, "%i", seNum);
    sendto(csock, temp, strlen(temp), 0, (struct sockaddr *)&si_other, slen);
    memset(temp, 0, sizeof(temp));
    Sleep(5);
```

```
    n = recvfrom(csock, temp, size, 0, (struct sockaddr *)&si_other, &slen);
    memset(temp, 0, sizeof(temp));
    Sleep(5);

    for (i = 0; i<seNum; i++){
           if (goods[i].id != 0){
                  sprintf(temp, "%i       %i       %i       %s", goods[i].id, goods[i].amount,
goods[i].price, goods[i].name);
                  sendto(csock, temp, strlen(temp), 0, (struct sockaddr *)&si_other, slen);
                  memset(temp, 0, sizeof(temp));
           }
    }
}

void admDial(btClient client){
    int n, com = 0;
    char temp[size];
    int csock = client.sockfd;
    struct sockaddr_in si_other = client.client;
    int slen = client.clilen;
    Sleep(5);
    n = recvfrom(csock, temp, size, 0, (struct sockaddr *) &si_other, &slen);
    temp[n] = 0;
    printf("recv=%s", temp);
    memset(temp, 0, sizeof(temp));
    Sleep(5);

    strcpy(temp, "\nAvailable commands : increase,decrease,create,remove,showlist,exit");
    sendto(csock, temp, strlen(temp), 0, (struct sockaddr *) &si_other, slen);
    memset(temp, 0, sizeof(temp));
    Sleep(5);

    strcpy(temp, "\nEnter command:\n");
    sendto(csock, temp, strlen(temp), 0, (struct sockaddr *) &si_other, slen);
    memset(temp, 0, sizeof(temp));


    n = recvfrom(csock, temp, size, 0, (struct sockaddr *) &si_other, &slen);
    temp[n] = 0;
    if (!strcmp(temp, "increase\r\n"))
           com = 1;
    if (!strcmp(temp, "decrease\r\n"))
           com = 2;
    if (!strcmp(temp, "create\r\n"))
           com = 3;
    if (!strcmp(temp, "remove\r\n"))
           com = 4;
    if (!strcmp(temp, "showlist\r\n"))
           com = 5;
    if (!strcmp(temp, "exit\r\n"))
           com = 6;
    memset(temp, 0, sizeof(temp));
    switch (com) {
    case 1:
           printf("\nincrease\n");
           add(client);
           break;
    case 2:
           printf("\ndecrease\n");
           decr(client);
           break;
    case 3:
           printf("\ncreate\n");
           create(client);
```

```
            break;
        case 4:
            printf("\nremove\n");
            rem(client);
            break;
        case 5:
            printf("\nshowlist\n");
            showList(client);
            break;
        case 6:
            printf("\nexit\n");
            disc();
            break;
        default:

            strcpy(temp, "Incorrect entry.\n");
            sendto(csock, temp, strlen(temp), 0, (struct sockaddr*)&si_other, slen);
            memset(temp, 0, sizeof(temp));

    }
}

void usrDial(btClient client){
    int n, com = 0;
    char temp[size];
    Sleep(5);
    int csock = client.sockfd;
    struct sockaddr_in si_other = client.client;
    int slen = client.clilen;
    n = recvfrom(csock, temp, size, 0, (struct sockaddr *) &si_other, &slen);
    temp[n] = 0;
    printf("recv=%s", temp);
    memset(temp, 0, sizeof(temp));
    Sleep(5);

    strcpy(temp, "\nAvailable commands : buy,showlist,exit");
    sendto(csock, temp, strlen(temp), 0, (struct sockaddr *) &si_other, slen);
    memset(temp, 0, sizeof(temp));
    Sleep(5);

    strcpy(temp, "\nEnter command:\n");
    sendto(csock, temp, strlen(temp), 0, (struct sockaddr *) &si_other, slen);
    memset(temp, 0, sizeof(temp));


    n = recvfrom(csock, temp, size, 0, (struct sockaddr *) &si_other, &slen);
    temp[n] = 0;
    if (!strcmp(temp, "buy\r\n"))
            com = 1;
    if (!strcmp(temp, "showlist\r\n"))
            com = 2;
    if (!strcmp(temp, "exit\r\n"))
            com = 3;
    memset(temp, 0, sizeof(temp));
    switch (com) {
    case 1:
            printf("buy\n");
            buy(client);
            break;
    case 2:
            printf("showlist\n");
            showList(client);
            break;
    case 3:
            printf("exit\n");
```

```
            disc();
            break;
    default:

            strcpy(temp, "Incorrect entry.\n");
            sendto(csock, temp, strlen(temp), 0, (struct sockaddr *) &si_other, slen);
            memset(temp, 0, sizeof(temp));
            break;
    }
}


void servInit(){
    FILE *id, *names, *prices, *amount;
    id = fopen("id.txt", "r");
    names = fopen("names.txt", "r");
    prices = fopen("prices.txt", "r");
    amount = fopen("amount.txt", "r");
    int i = 0;
    while (!feof(names)) {
            fgets(goods[i].name, 35, names);
            fscanf(id, "%i", &goods[i].id);
            fscanf(prices, "%i", &goods[i].price);
            fscanf(amount, "%i", &goods[i].amount);
            i++;
            seNum++;
    }
    strcat(goods[seNum - 1].name, "\n");
    fclose(id);
    fclose(names);
    fclose(prices);
    fclose(amount);
}

unsigned _stdcall handler(void* param){
    char str[size];
    int n, usr = 0;

    btClient* cli = (btClient*)param;
    btClient client = *cli;
    int csock = client.sockfd;
    struct sockaddr_in si_other = client.client;
    int slen = client.clilen;

    if ((recvfrom(csock, str, size, 0, (struct sockaddr *) &si_other, &slen)) == SOCKET_ERROR)
    {
            printf("recvfrom() failed with error code : %d", WSAGetLastError());
            exit(EXIT_FAILURE);
    }
    memset(str, 0, sizeof(str));
    //print details of the client/peer and the data received
    Sleep(5);
    sendto(csock, "\n\nWelcome to our e-shop!\n", strlen("\n\nWelcome to our e-shop!\n"), 0, (struct
sockaddr *) &si_other, slen);
    Sleep(5);
    sendto(csock, "Enter your name:\n", strlen("Enter your name:\n"), 0, (struct sockaddr *)
&si_other, slen);
    n = recvfrom(csock, str, size, 0, (struct sockaddr *) &si_other, &slen);
    str[n] = 0;

    if (!strcmp(str, "admin")){
            usr = 1;
    }
    else
            usr = 2;
```

```
    while (1){
           switch (usr) {
           case 1:
                  admDial(client);
                  break;
           case 2:
                  usrDial(client);
                  break;
           }
           if (cl != 0){
                  cl = 0;
                  break;
           }
    }
    memset(str, 0, sizeof(str));
    strcpy(str, "\nConnection closed.\n");
    sendto(csock, str, strlen("\nConnection closed.\n"), 0, (struct sockaddr *) &si_other, slen);
    memset(str, 0, sizeof(str));
    closesocket(csock);
};

int main()
{
    SOCKET s;
    struct sockaddr_in si_other;
    int slen = sizeof(si_other);
    struct sockaddr_in server;
    int n;
    int newport;
    char str[size];
    WSADATA wsa;

    servInit();
    //Initialise winsock
    printf("\nInitialising Winsock...");
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
    {
           printf("Failed. Error Code : %d", WSAGetLastError());
           exit(EXIT_FAILURE);
    }
    printf("Initialised.\n");

    //Create a socket
    if ((s = socket(AF_INET, SOCK_DGRAM, 0)) == INVALID_SOCKET)
    {
           printf("Could not create socket : %d", WSAGetLastError());
    }
    printf("Socket created.\n");

    //Prepare the sockaddr_in structure
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = htons(PORT);

    //Bind
    if (bind(s, (struct sockaddr *)&server, sizeof(server)) == SOCKET_ERROR)
    {
           printf("Bind failed with error code : %d", WSAGetLastError());
           exit(EXIT_FAILURE);
    }
    puts("Bind done");

    while (1){
           int ns;
           recvfrom(s, str, size, 0, (struct sockaddr *) &si_other, &slen);
```

```
            struct sockaddr_in serveraddr;
            ns = socket(AF_INET, SOCK_DGRAM, 0);
            clientsNum++;
            newport = PORT + clientsNum + 1;
            sprintf(str, "%i", newport);
            printf("%i\n", newport);

            int optval = 1;
            setsockopt(ns, SOL_SOCKET, SO_REUSEADDR, (char *)&optval, sizeof(int));


            memset((char *)&serveraddr, '\0', sizeof(serveraddr));

            serveraddr.sin_family = AF_INET;
            serveraddr.sin_addr.s_addr = INADDR_ANY;
            serveraddr.sin_port = htons(newport);
            bind(ns, (struct sockaddr *)&serveraddr, sizeof(serveraddr));

            sendto(s, str, strlen(str), 0, (struct sockaddr *)&si_other, slen);
            memset(str, 0, sizeof(str));

            struct sockaddr_in newclientaddr; /* client addr */
            int newclientlen = sizeof(newclientaddr); /* byte size of client's address */
            memset(str, 0, size);
            recvfrom(ns, str, sizeof(str), 0, (struct sockaddr *) &newclientaddr, &newclientlen);
            printf("\ngg=%s\n", str);

            btClient client;
            client.sockfd = ns;
            client.client = newclientaddr;
            client.clilen = newclientlen;

            HANDLE t;
            t = (HANDLE)_beginthreadex(NULL, 0, &handler, (void*)&client, 0, NULL);
    }
    closesocket(s);
    WSACleanup();
}
```

UDP клиент

```
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <fcntl.h>
#define size 1024
#define BUFLEN 1024
#define NPACK 10
#define PORT 8888

typedef struct {
    int sockfd;
    struct sockaddr_in client;
    int clilen;
} btClient;
int cl = 0;
```

```
void rec(char *str1, btClient *client){
    int n;
    n = recvfrom(client->sockfd, str1, size, 0, (struct sockaddr *)&client->client, &client-
>clilen);
    str1[n] = 0;
}

void showList(btClient *client){
    char str[size];
    int counter, i;
    int sock = client->sockfd;
    struct sockaddr_in si_other = client->client;
    int slen = client->clilen;

    sendto(sock, "1", strlen("1"), 0, (struct sockaddr *)&si_other, slen);
    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));

    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));
    rec(str, client);
    counter = atoi(str);
    memset(str, 0, sizeof(str));

    sendto(sock, "1", strlen("1"), 0, (struct sockaddr *)&si_other, slen);

    for (i = 0; i < counter - 1; i++){
           rec(str, client);
           printf(str);
           memset(str, 0, sizeof(str));
    }
}


void increase(btClient *client){
    char str[size];

    int sock = client->sockfd;
    struct sockaddr_in si_other = client->client;
    int slen = client->clilen;


    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    sendto(sock, str, strlen(str), 0, (struct sockaddr *)&si_other, slen);
    memset(str, 0, sizeof(str));


    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    sendto(sock, str, strlen(str), 0, (struct sockaddr *)&si_other, slen);
    memset(str, 0, sizeof(str));
```

```
    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));
}


void decrease(btClient *client){
    char str[size];

    int sock = client->sockfd;
    struct sockaddr_in si_other = client->client;
    int slen = client->clilen;

    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    sendto(sock, str, strlen(str), 0, (struct sockaddr *)&si_other, slen);
    memset(str, 0, sizeof(str));


    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    sendto(sock, str, strlen(str), 0, (struct sockaddr *)&si_other, slen);
    memset(str, 0, sizeof(str));



    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));


    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));
}


void create(btClient *client){
    char str[size];

    int sock = client->sockfd;
    struct sockaddr_in si_other = client->client;
    int slen = client->clilen;

    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\n");
    sendto(sock, str, strlen(str), 0, (struct sockaddr *)&si_other, slen);
    memset(str, 0, sizeof(str));



    rec(str, client);
    printf(str);
```

```
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    sendto(sock, str, strlen(str), 0, (struct sockaddr *)&si_other, slen);
    memset(str, 0, sizeof(str));




    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    sendto(sock, str, strlen(str), 0, (struct sockaddr *)&si_other, slen);
    memset(str, 0, sizeof(str));
}


void rem(btClient *client){
    char str[size];

    int sock = client->sockfd;
    struct sockaddr_in si_other = client->client;
    int slen = client->clilen;


    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");
    sendto(sock, str, strlen(str), 0, (struct sockaddr *)&si_other, slen);
    memset(str, 0, sizeof(str));


    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));
}


void admDial(btClient *client){
    char str[size];
    int com = 0;

    int sock = client->sockfd;
    struct sockaddr_in si_other = client->client;
    int slen = client->clilen;
    sendto(sock, "1", strlen("1"), 0, (struct sockaddr *)&si_other, slen);



    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));


    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));
```

```
    scanf("%s", str);
    strcat(str, "\r\n");

    if (!strcmp(str, "increase\r\n")){
            com = 1;
            sendto(sock, "increase\r\n", strlen("increase\r\n"), 0, (struct sockaddr *)&si_other,
slen);
            increase(client);
    }
    if (!strcmp(str, "decrease\r\n")){
            com = 2;
            sendto(sock, "decrease\r\n", strlen("decrease\r\n"), 0, (struct sockaddr *)&si_other,
slen);
            decrease(client);
    }
    if (!strcmp(str, "create\r\n")){
            com = 3;
            sendto(sock, "create\r\n", strlen("create\r\n"), 0, (struct sockaddr *)&si_other, slen);
            create(client);
    }

    if (!strcmp(str, "remove\r\n")){
            com = 4;
            sendto(sock, "remove\r\n", strlen("remove\r\n"), 0, (struct sockaddr *)&si_other, slen);
            rem(client);
    }
    if (!strcmp(str, "showlist\r\n")){
            com = 5;
            sendto(sock, "showlist\r\n", strlen("showlist\r\n"), 0, (struct sockaddr *)&si_other,
slen);
            showList(client);
    }
    if (!strcmp(str, "exit\r\n")){
            com = 6;
            sendto(sock, "exit\r\n", strlen("exit\r\n"), 0, (struct sockaddr *)&si_other, slen);
            cl = 1;
    }
    if (com == 0){

            sendto(sock, "incorrect\r\n", strlen("incorrect\r\n"), 0, (struct sockaddr *)&si_other,
slen);


            rec(str, client);
            printf(str);
            memset(str, 0, sizeof(str));
    }
    memset(str, 0, sizeof(str));
}



void usrDial(btClient *client){
    char str[size];
    int com = 0;
    int sock = client->sockfd;
    struct sockaddr_in si_other = client->client;
    int slen = client->clilen;

    sendto(sock, "1", strlen("1"), 0, (struct sockaddr *)&si_other, slen);



    rec(str, client);
    printf(str);
```

```
    memset(str, 0, sizeof(str));


    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));

    scanf("%s", str);
    strcat(str, "\r\n");

    if (!strcmp(str, "showlist\r\n")){
           com = 1;
           sendto(sock, "showlist\r\n", strlen("showlist\r\n"), 0, (struct sockaddr *)&si_other,
slen);
           showList(client);
    }
    if (!strcmp(str, "buy\r\n")){
           com = 2;
           sendto(sock, "buy\r\n", strlen("buy\r\n"), 0, (struct sockaddr *)&si_other, slen);
           decrease(client);
    }
    if (!strcmp(str, "exit\r\n")){
           com = 3;
           sendto(sock, "exit\r\n", strlen("exit\r\n"), 0, (struct sockaddr *)&si_other, slen);
           cl = 1;
    }
    if (com == 0){

           sendto(sock, "incorrect\r\n", strlen("incorrect\r\n"), 0, (struct sockaddr *)&si_other,
slen);


           rec(str, client);
           printf(str);
           memset(str, 0, sizeof(str));
    }
    memset(str, 0, sizeof(str));
}

void handler(void* param){
    char str[BUFLEN];
    char inp[BUFLEN];
    int usr = 0, i;

    btClient* client = (btClient*)param;
    int sock = client->sockfd;
    struct sockaddr_in si_other = client->client;
    int slen = client->clilen;

    sendto(sock, "1", strlen("1"), 0, (struct sockaddr *) &si_other, slen);
    for (i = 0; i<2; i++) {

           rec(str, client);
           printf(str);
           memset(str, 0, sizeof(str));
    }
    scanf("%s", inp);
    sendto(sock, inp, strlen(inp), 0, (struct sockaddr *) &si_other, slen);
    if (!strcmp(inp, "admin")){
           usr = 1;
    }
    else
           usr = 0;
    while (cl == 0){
           switch (usr) {
```

```
            case 1:
                    memset(inp, 0, sizeof(inp));
                    admDial(client);
                    break;
            case 0:
                    memset(inp, 0, sizeof(inp));
                    usrDial(client);
                    break;
            }
    }

    rec(str, client);
    printf(str);
    memset(str, 0, sizeof(str));
    close(sock);
}


int main(void)
{
    struct sockaddr_in si_other;
    int slen = sizeof(si_other);
    int s, n, newport;
    char str[size];
    char SRV_IP[size];
    printf("Enter servers's ip:\n");
    scanf("%s", SRV_IP);
    //struct sockaddr_in si_other;
    memset((char *)&si_other, 0, sizeof(si_other));
    si_other.sin_family = AF_INET;
    si_other.sin_port = htons(PORT);
    if (inet_aton(SRV_IP, &si_other.sin_addr) == 0) {
            fprintf(stderr, "inet_aton() failed\n");
            exit(1);
    }
    if ((s = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1)
            perror("socket");
    memset(str, 0, sizeof(str));

    sendto(s, "1", strlen("1"), 0, (struct sockaddr *) &si_other, slen);

    n = recvfrom(s, str, size, 0, (struct sockaddr *)&si_other, &slen);
    str[n] = 0;
    newport = atoi(str);
    printf("port=%i", newport);
    memset(str, 0, sizeof(str));
    struct sockaddr_in new_si_other;
    int ns, new_slen = sizeof(new_si_other);

    ns = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    //fcntl(ns, F_SETFL, O_NONBLOCK);
    memset((char *)&new_si_other, 0, new_slen);
    new_si_other.sin_family = AF_INET;
    new_si_other.sin_port = htons(newport);
    inet_aton(SRV_IP, &new_si_other.sin_addr);

    btClient client;
    client.sockfd = ns;
    client.client = new_si_other;
    client.clilen = sizeof(new_si_other);

    close(s);
    sendto(client.sockfd, "newclient created", strlen("newclient created"), 0, (struct sockaddr *)
&client.client, client.clilen);
    handler((void*)&client);
```

```
}
```