Санкт-Петербургский государственный политехнический университет

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе

«Технологии компьютерных сетей»

Игра «Рулетка»

Работу выполнил студент группы 43501/3
Кенть Н.В.

Работу приняла Вылегжанина К.Д.

Санкт-Петербург

2014

Глава 1 Задание

Разработать клиент- серверное приложение «Рулетка».

Рулетка – азартная игра, в которой большой шанс выигрыша за счет того, что для победы достаточно угадать четное ли выпадет число. Игрок пытается угадать номер, который выпадет и его четность\нечетность. Наша задача разработать клиент- серверное приложение для этой игры, используя протоколы TCP, UDP

1.1 Функциональные требования

Серверное приложение реализовывает следующие функции:

- Регистрация подключившегося клиента в качестве крупье (один) или в качестве игрока (несколько)
- Выдача клиенту (крупье и игроку) списка ставок текущей игры (ставок других игроков)
- Получение от клиента ставки: суммы ставки и типа ставки (чет, нечет, номер)
- Получение от крупье команды на начало розыгрыша.
- Уведомление всех клиентов о результате розыгрыша

Клиентское приложение реализовывает следующие функции:

- Посылка регистрационных данных клиента (как крупье или как игрока)
- Получение и вывод списка ставок
- Для игрока: посылка своей ставки (сумма и ставки и тип ставки)
- Для крупье: посылка команды начала розыгрыша
- Получение и вывод результатов розыгрыша

1.2 Нефункциональные требования

Нефункциональные требования

Серверное приложение:

- Прослушивание определенного порта
- Обработка запросов на подключение по этому порту от клиентов
- Поддержка одновременной работы нескольких клиентов через механизм нитей

Клиентское приложение должно устанавливать соединение с сервером.

1.2 Накладываемы ограничения

- Игрок может делать ставку от 1000 до 9999
- Игрок может выбрать номер от 00 до 99
- Игрок может выбрать четное или нечетное.

Глава 2

Реализация для работы по протоколу TCP

2.1 Общая информация

TCP (англ. *Transmission Control Protocol*, протокол управления передачей) — один из основных протоколов передачи данных Интернета, предназначенный для управления передачей данных в сетях и подсетях TCP/IP.

Выполняет функции протокола транспортного уровня в стеке протоколов IP.

Механизм TCP предоставляет поток данных с предварительной установкой соединения, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета, гарантируя тем самым, в отличие от UDP, целостность передаваемых данных и уведомление отправителя о результатах передачи.

2.2 Прикладной протокол

Сервер принимает от клиентов запросы на игру формата: x;y;z, Где,

x (0 или 1) - запрос на игру, 1 - хочу играть.

y - роль : 0- крупье, 1= игрок,

z - имя.

Сервер присылает сообщение с ролью клиента, если выбранную роль невозможно выбрать, то будет соответствующее сообщение.

Затем игроки вводят ставки

Ввод ставок: x;y;z-

x- номер (вид:00, 01, 02,...,99),

y- 0 (чет) или 1(нечет),

z- сумма ставки (от 1000 до 9999).

Сервер отвечает, что ставка принята.

Начало розыгрыша.

После приёма ставок сервер принимает от крупье запрос на выдачу ставок. Формат : 1.

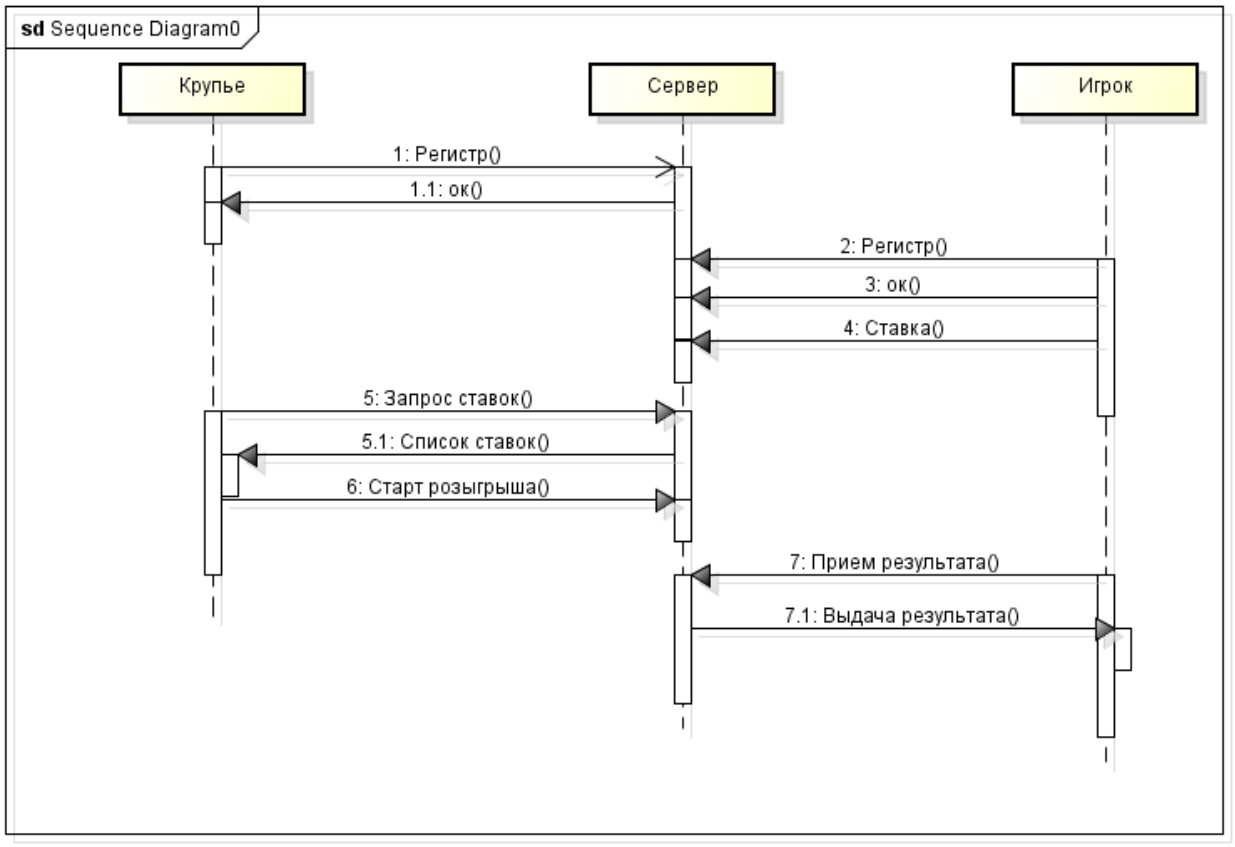
Розыгрыш.

Крупье подает сигнал старта розыгрыша: s, а игроки сигнал приёма результатов: r.

Все ставки, имена и результаты розыгрыша хранятся в файлах, доступ к которым есть только на сервере.

Таблица команд клиента с описанием.

Команда	Пример ввода	Описание
Вход клиента	0;0;Crop	Клиент говорит, что хочет играть как крупье, его имя –Croup
Ввод ставок (игрок)	12;0;5400	Игрок ставит на номер 12, четное. Сумма ставки 5400
Выдача списка ставок (крупье)	1	Крупье делает запрос на выдачу сделанных ставок
Старт розыгрыша (крупье)	s	Крупье даёт команду на старт. Генерируется победная комбинация
Прием результата (игрок)	r	Игрок дает команду и сервер присылает ему его результаты



Sequence diagram

2.3 Архитектура приложения

Клиент начинает общение с сервером.

В игре существует два вида клиентов – крупье и игрок. Первый подключившийся, желающий быть крупье становится таковым. Затем подключаются игроки. Они делают ставки. Крупье дает команду на получение ставок, затем на старт розыгрыша. Игроки дают команду на получение результатов.

На сервере хранится 3 файла. Первый файл – файл с данными игроков (имена, роли). Файл 2 – ставки игроков. Файл 3 – результат розыгрыша (Победная комбинация).

При подключении нового клиента, если его запрос корректный, сервер заносит его данные в первый файл. Если клиент – игрок, то его ставка заносится во второй файл. После команды крупье о старте розыгрыша сервер заносит победный номер в третий файл.

2.4 Тестирование

Взаимодействие серверного и клиентского приложения проверялось на одном компьютере. В одном терминале запускался сервер, в других клиенты.

При вводе правильных команд приложение работает корректно с несколькими клиентами.

В случае, если клиент захочет быть игроком, но крупье еще нет будет выведено сообщение о том, что крупье еще нет. Если клиент захочет стать вторым крупье, то также появится уведомление, что это не возможно. Подключение нескольких клиентов не меняет условия розыгрыша. Для каждого клиента подводится результат на основании его ставки.

Так как мы подразумеваем, что протокол известен и формат команд известен для клиента, то мы не делаем проверку корректности ввода команд.

После розыгрыша для корректной игры требуется очистить файлы на сервере. Для этого был написан скрипт. После каждого розыгрыша нужно вручную запустить скрипт.

Глава 3

Реализация для работы по протоколу UDP

3.1 Общая информация

UDP (англ. *User Datagram Protocol* — протокол пользовательских дэйтаграмм) — один из ключевых элементов TCP/IP, набора сетевых протоколов для Интернета. С UDP компьютерные приложения могут посылать сообщения (в данном случае называемые дэйтаграммами) другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных.

3.2 Прикладной протокол

Протокол не понёс изменения. Описание см. п 2.2.

3.3 Многоклиентское взаимодействие.

Сложность реализации многоклиентского взаимодействия состоит в том, что по протоколу UDP клиент не устанавливает соединения с сервером, из-за этого не получилось реализовать поддержку нескольких клиентов. Была попытка реализовать приложение логически так, чтобы использовать особенность того, что сервер принимает сообщение от всех клиентов, однако, из-за того, что нужно поддерживать цепочку событий, которая может развиваться по- разному это не удалось.

3.4 Тестирование UDP- приложения

Так как логика приложения не изменилась, тестирование аналогично TCP версии, но без поддержки многопоточности. Один клиент работает корректно (крупье или игрок).

Выводы:

Когда осуществляется передача от компьютера к компьютеру через Интернет, TCP работает на верхнем уровне между двумя конечными системами, например, браузером и веб-сервером. TCP осуществляет надежную передачу потока байтов от одной программы на некотором компьютере к другой программе на другом компьютере (например, программы для электронной почты, для обмена файлами). TCP контролирует длину сообщения, скорость обмена сообщениями, сетевой трафик.

UDP использует простую модель передачи, без неявных «рукопожатий» для обеспечения надёжности, упорядочивания или целостности данных. Таким образом, UDP предоставляет ненадёжный сервис, и датаграммы могут прийти не по порядку, дублироваться или вовсе исчезнуть без следа. UDP подразумевает, что проверка ошибок и исправление либо не нужны, либо должны исполняться в приложении.

Для данного приложения, я считаю, что более подходит протокол TCP, так как он обеспечивает надежность передачи, мы устанавливаем соединение с клиентом, поэтому просто организовывать многоклиентское взаимодействие. Скорость передачи нам не очень важна, гораздо важнее надежность.

Исходный код

Код udp_server_linux:

```
/* Sample UDP server */


#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <pthread.h>
#include <netdb.h>
pthread_t t1, t2, t3;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
struct sockaddr_in servaddr,
cliaddr;
void thread2(int sock) {
        char buffer[256];
        int N = 256;
        int n;
        char mesg[256];


        char buffer_file[256];


        //      int n;
        int c;
        int i = 0;
        int player;
```

```
int number, odd;
char my_number[3];
char my_odd_or_even;
char my_bet[5];
bzero(buffer, N);
socklen_t len;
len = sizeof(cliaddr);
/*      n = recvfrom(sock,buffer,N,0,(struct sockaddr *)&cliaddr,&len);
 sendto(sock,buffer,N,0,(struct sockaddr *)&cliaddr,sizeof(cliaddr));
 printf("--------------------------------------------------------\n");
 //  buffer[n] = 0;
 printf("Received the following:\n");
 printf("%s",buffer);
 printf("--------------------------------------------------------\n");*/
n = recvfrom(sock, buffer, N, 0, (struct sockaddr *) &cliaddr, &len);
printf("Here is the message: %s\n", buffer);
FILE * fo;
fo = fopen("test.txt", "r");
bzero(buffer_file, N);
fgets(buffer_file, 50, fo);

fclose(fo);

if (buffer_file[2] == '0') {
        if (buffer[2] == '1') {
```

```
                fo = fopen("test.txt", "a+");
                fputs("\n", fo);
                fputs(buffer, fo);
                fclose(fo);
                player = 1;
                //      write(sock, "You are in game, player\n", 24);
                sendto(sock, "You are in game, player\n", N, 0,
                                (struct sockaddr *) &cliaddr, sizeof(cliaddr));
        } else
                //              write(sock, "u cant be a croup\n", 18);
                sendto(sock, "u cant be a croup\n", N, 0,
                                (struct sockaddr *) &cliaddr, sizeof(cliaddr));
} else {
        if (buffer[2] == '0') {
                fo = fopen("test.txt", "a+");

                fputs(buffer, fo);
                fclose(fo);
                player = 0;
                //write(sock, "You are in game, croup\n", 23);
                sendto(sock, "You are in game, croup\n", N, 0,
                                (struct sockaddr *) &cliaddr, sizeof(cliaddr));
        } else {
                //      write(sock, "u cant be a player", 18);
                sendto(sock, "u cant be a player", N, 0,
```

```
                        (struct sockaddr *) &cliaddr, sizeof(cliaddr));
        }


}
bzero(buffer, N);
if (player == 0)
        sendto(sock, "waiting...players are making bets\n", N, 0,
                        (struct sockaddr *) &cliaddr, sizeof(cliaddr));
else
        sendto(sock, "make your bet", N, 0, (struct sockaddr *) &cliaddr,
                        sizeof(cliaddr));
//n = read(sock, buffer, N - 1);
n = recvfrom(sock, buffer, N, 0, (struct sockaddr *) &cliaddr, &len);
printf("Here is the message: buffer[0]= %c, player= %d\n", buffer[0],
                player);
if (player == 0) {
        if (buffer[0] == "1")
                printf("its time to show bets\n");
        fo = fopen("bets.txt", "r");
        while (fgets(buffer, 50, fo) != NULL) {
                //              printf("bets: %s\n", buffer);
                //      write(sock, buffer, strlen(buffer));
                sendto(sock, buffer, N, 0, (struct sockaddr *) &cliaddr,
                                sizeof(cliaddr));
        }
```

```
} else {
        //write(sock, "i got your bet", 14);
        sendto(sock, "i got your bet", N, 0, (struct sockaddr *) &cliaddr,
                        sizeof(cliaddr));
        fo = fopen("bets.txt", "a+");
        fputs(buffer, fo);
        fputs("\n", fo);
        fclose(fo);
        while (buffer[i] != ';') {
                my_number[i] = buffer[i];
                i++;
        }
        i++;
        my_odd_or_even = buffer[i];
        i++;
        int j = 0;
        for (i = 5; i < 10; i++, j++) {
                my_bet[j] = buffer[i];

        }

}

if (player == 0) {
        bzero(buffer, N);
```

```
        //n = read(sock, buffer, N - 1);
        n = recvfrom(sock, buffer, N, 0, (struct sockaddr *) &cliaddr, &len);
        if (buffer[0] == 's') {
                srand(time(NULL));
                number = rand() % 100;
                odd = rand() % 2;
                fo = fopen("game.txt", "a+");
                fprintf(fo, "%d", number);
                fputs("\n", fo);
                fprintf(fo, "%d", odd);
                fclose(fo);
        }

}
int win1 = 5;
int win2 = 5;
if (player == 1) {
        //n = read(sock, buffer, N - 1);
        n = recvfrom(sock, buffer, N, 0, (struct sockaddr *) &cliaddr, &len);
        if (buffer[0] == 'r') {
                printf("new check\n");

                fo = fopen("game.txt", "r");
                fgets(buffer, 50, fo);
```

```
                    if (strcmp(buffer, my_number) == 0)
                            win1 = 1;

                    else
                            win1 = 0;
                    fgets(buffer, 50, fo);
                    if (buffer[0] == my_odd_or_even)
                            win2 = 1;

                    else
                            win2 = 0;

                    fclose(fo);
                    if (win1 == 1 || win2 == 1)
                            //      write(sock, "Granz, you won", 14);
                            sendto(sock, "Granz, you won", N, 0,
                                    (struct sockaddr *) &cliaddr,
sizeof(cliaddr));
                    else
                            //      write(sock, "you loose", 9);
                            sendto(sock, "you loose", N, 0, (struct sockaddr *)
&cliaddr,
                                    sizeof(cliaddr));
                }
        }
}
```

```
struct sockaddr_in servaddr, cliaddr;
/*static void test(void* arg) {
 pthread_mutex_lock(&mutex);

 int sock = *(int*) arg;
 char mesg[1000];
 socklen_t len;
 int n;
 char buffer[256];
 int N = 256;

 char buffer_file[256];

 //       int n;
 int c;
 int i = 0;
 int player;
 int number, odd;
 char my_number[3];
 char my_odd_or_even;
 char my_bet[5];
 for (;;) {
 /*                len = sizeof(cliaddr);
 n= recvfrom(sockfd,mesg,1000,0,(struct sockaddr *)&cliaddr,&len);
 mesg[n] = 0;
```

```
printf("%s",mesg);
sendto(sockfd,"hekko",10,0,(struct sockaddr *)&cliaddr,sizeof(cliaddr));*/
/*
bzero(buffer, N);

n = recvfrom(sock, buffer, N, 0, (struct sockaddr *) &cliaddr, &len);
printf("Here is the message: %s\n", buffer);
FILE * fo;
fo = fopen("test.txt", "r");
bzero(buffer_file, N);
fgets(buffer_file, 50, fo);

fclose(fo);

if (buffer_file[2] == '0') {
if (buffer[2] == '1') {
fo = fopen("test.txt", "a+");
fputs("\n", fo);
fputs(buffer, fo);
fclose(fo);
player = 1;
//      write(sock, "You are in game, player\n", 24);
sendto(sock, "You are in game, player\n", N, 0,
(struct sockaddr *) &cliaddr, sizeof(cliaddr));
} else
```

```
//                write(sock, "u cant be a croup\n", 18);
sendto(sock, "u cant be a croup\n", N, 0,
(struct sockaddr *) &cliaddr, sizeof(cliaddr));
} else {
if (buffer[2] == '0') {
fo = fopen("test.txt", "a+");

fputs(buffer, fo);
fclose(fo);
player = 0;
//write(sock, "You are in game, croup\n", 23);
sendto(sock, "You are in game, croup\n", N, 0,
(struct sockaddr *) &cliaddr, sizeof(cliaddr));
} else {
//       write(sock, "u cant be a player", 18);
sendto(sock, "u cant be a player", N, 0,
(struct sockaddr *) &cliaddr, sizeof(cliaddr));
}

}
bzero(buffer, N);
if (player == 0)
sendto(sock, "waiting...players are making bets\n", N, 0,
(struct sockaddr *) &cliaddr, sizeof(cliaddr));
else
```

```
sendto(sock, "make your bet", N, 0, (struct sockaddr *) &cliaddr,
sizeof(cliaddr));
n = recvfrom(sock, buffer, N, 0, (struct sockaddr *) &cliaddr, &len);
printf("Here is the message: buffer[0]= %c, player= %d\n", buffer[0],
player);

if (player == 0) {
if (buffer[0] == "1")
printf("its time to show bets\n");
fo = fopen("bets.txt", "r");
while (fgets(buffer, 50, fo) != NULL) {
//                printf("bets: %s\n", buffer);
//      write(sock, buffer, strlen(buffer));
sendto(sock, "bets made", N, 0, (struct sockaddr *) &cliaddr,
sizeof(cliaddr));
}
} else {
//write(sock, "i got your bet", 14);
sendto(sock, "i got your bet", N, 0, (struct sockaddr *) &cliaddr,
sizeof(cliaddr));
fo = fopen("bets.txt", "a+");
fputs(buffer, fo);
fputs("\n", fo);
fclose(fo);
while (buffer[i] != ';') {
```

```
my_number[i] = buffer[i];
i++;
}
i++;
my_odd_or_even = buffer[i];
i++;
int j = 0;
for (i = 5; i < 10; i++, j++) {
my_bet[j] = buffer[i];

}

}
}

pthread_mutex_unlock(&mutex);
shutdown(sock, 2);
close(sock);
}*/

int main(int argc, char**argv) {
        int sockfd, n;

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);
        pthread_t accept_thread;
```

```
	bzero(&servaddr, sizeof(servaddr));
	servaddr.sin_family = AF_INET;
	servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
	servaddr.sin_port = htons(32000);
	int optval = 1;
	setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &optval, sizeof
optval);
	bind(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr));
	thread2(sockfd);
}
```

UDP_Client_linux:

```
/* Sample UDP client */

#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>

int main(int argc, char**argv) {
	int sockfd, n;
	struct sockaddr_in servaddr, cliaddr;
	char sendline[1000];
	char recvline[1000];

	if (argc != 2) {
		printf("usage:  udpcli <IP address>\n");
```

```
        exit(1);
}

sockfd = socket(AF_INET, SOCK_DGRAM, 0);

bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = inet_addr(argv[1]);
servaddr.sin_port = htons(32000);
char buffer[256];
scanf("%s", buffer);
sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr *) &servaddr,
                sizeof(servaddr));
//    getchar ();
n = recvfrom(sockfd, buffer, 10000, 0, NULL, NULL);
//    buffer[n]='\0';
printf("%s\n", buffer);
bzero(buffer, 256);
n = recvfrom(sockfd, buffer, 10000, 0, NULL, NULL);
//    buffer[n]='\0';
printf("%s\n", buffer);

scanf("%s", buffer);
sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr *) &servaddr,
                sizeof(servaddr));
```

```
bzero(buffer, 256);
n = recvfrom(sockfd, buffer, 10000, 0, NULL, NULL);
//    buffer[n]='\0';
printf("%s\n", buffer);
bzero(buffer, 256);
scanf("%s", buffer);
sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr *) &servaddr,
                    sizeof(servaddr));
//start from croup

bzero(buffer, 256);
n = recvfrom(sockfd, buffer, 256, 0, NULL, NULL);

printf("%s\n", buffer);
/*        bzero(buffer, 256);
 scanf("%s", buffer);
 sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr *) &servaddr,
 sizeof(servaddr));
 bzero(buffer, 256);
 n = recvfrom(sockfd, buffer, 256, 0, NULL, NULL);
 printf("%s\n", buffer);

 bzero(buffer, 256);
 scanf("%s", buffer);
 sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr *) &servaddr,
```

```
        sizeof(servaddr));
        bzero(buffer, 256);
        n = recvfrom(sockfd, buffer, 256, 0, NULL, NULL);
        printf("%s\n", buffer);*/

}
```

TCP_server_linux

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdlib.h>
//#include <stdio.h>
//#include <stdlib.h>
//#include <unistd.h>
#include <string.h>
#include <time.h>
//#include <sys/types.h>
//#include <sys/socket.h>
//include <netinet/in.h>
#include <netdb.h>
#include <pthread.h>
pthread_t t1, t2, t3;
```

```
void* thread2(int sock) {
        char buffer[256];
        char buffer_file[256];
        int N = 256;
        int n;
        int c;
        int i = 0;
        int player;
        int number, odd;
        char my_number [3];
        char my_odd_or_even;
        char my_bet [5];
        bzero(buffer, N);

        n = read(sock, buffer, N - 1);
        printf("Here is the message: %s\n", buffer);
        FILE * fo;

        fo = fopen("test.txt", "r");
        bzero(buffer_file, N);
        fgets(buffer_file, 50, fo);

        fclose(fo);
```

```
if (buffer_file[2] == '0') {
        if (buffer[2] == '1') {
                fo = fopen("test.txt", "a+");
                fputs("\n", fo);
                fputs(buffer, fo);
                fclose(fo);
                player = 1;
                write(sock, "You are in game, player\n", 24);
        } else
                write(sock, "u cant be a croup\n", 18);
} else {
        if (buffer[2] == '0') {
                fo = fopen("test.txt", "a+");

                fputs(buffer, fo);
                fclose(fo);
                player = 0;
                write(sock, "You are in game, croup\n", 23);
        } else {
                write(sock, "u cant be a player", 18);
        }

}
bzero(buffer, N);
if (player == 0)
```

```
        write(sock, "waiting...players are making bets\n", 34);
else
        write(sock, "make your bet", 13);


n = read(sock, buffer, N - 1);
printf("Here is the message: buffer[0]= %c, player= %d\n", buffer[0],
                player);
if (player == 0) {
        if (buffer[0] == "1")
                printf("its time to show bets\n");
        fo = fopen("bets.txt", "r");
        while (fgets(buffer, 50, fo) != NULL) {
//              printf("bets: %s\n", buffer);
                write(sock, buffer, strlen(buffer));
        }
        write(sock,"1", 1);
} else {
        write(sock, "i got your bet", 14);
        write(sock,"1", 1);
        fo = fopen("bets.txt", "a+");
        fputs(buffer, fo);
        fputs("\n", fo);
        fclose(fo);
        while (buffer[i]!=';') {
                my_number[i]=buffer[i];
```

```
                        i++;
        }
        i++;
        my_odd_or_even=buffer[i];
        i++;
        int j=0;
        for(i=5;i <10;i++,j++) {
                my_bet[j]=buffer[i];

        }

}

if (player == 0) {
        bzero(buffer, N);
        n = read(sock, buffer, N - 1);
        if (buffer[0] == 's') {
                srand(time(NULL));
                number = rand() % 100;
                odd = rand() % 2;
                fo = fopen("game.txt", "a+");
                fprintf(fo, "%d", number);
                fputs("\n", fo);
                fprintf(fo, "%d", odd);
                fclose(fo);
```

```
        }


}
int win1=5;
int win2=5;
if (player == 1) {
        n = read(sock, buffer, N - 1);
        if (buffer[0] == 'r') {
                printf("new check\n");


                fo = fopen("game.txt", "r");
                fgets(buffer, 50, fo);


                if (strcmp(buffer,my_number)==0)
                        win1=1;


                else
                        win1=0;
                fgets(buffer, 50, fo);
                if (buffer[0]==my_odd_or_even)
                                        win2=1;


                                else
                                        win2=0;
```

```
                        fclose(fo);
                        if (win1==1||win2==1)
                                write(sock, "Granz, you won", 14);
                        else
                                write(sock, "you loose", 9);
                }
        }

}
void* thread1(int sock) {
        write(sock, "its thread1", 11);
//      printf("its thread\n");
}
int main(int argc, char *argv[]) {
        int sockfd, newsockfd, portno, clilen, client;
        char buffer[256];
        struct sockaddr_in serv_addr, cli_addr;
        int n;

        /* First call to socket() function */
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd < 0) {
                perror("ERROR opening socket");
                exit(1);
        }
```

```
/* Initialize socket structure */
bzero((char *) &serv_addr, sizeof(serv_addr));
portno = 5001;
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(portno);
const int on = 1;
setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on));
/* Now bind the host address using bind() call.*/
if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
        perror("ERROR on binding");
        exit(1);
}

/* Now start listening for the clients, here process will
 * go in sleep mode and will wait for the incoming connection
 */

listen(sockfd, 5);
clilen = sizeof(cli_addr);
bzero(buffer, 256);

while (1) {

        newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
```

```
            if (newsockfd < 0) {
                perror("ERROR on accept");
                exit(1);
            }

            pthread_create(&t2, NULL, thread2, newsockfd);

        }
        pthread_join(t2, NULL);
        return 0;
}
```

Tcp_client_linux

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

int main(int argc, char *argv[])
{
   int sockfd, portno, n;
```

```
struct sockaddr_in serv_addr;
struct hostent *server;

char buffer[256];

if (argc < 3) {
   fprintf(stderr,"usage %s hostname port\n", argv[0]);
   exit(0);
}
portno = atoi(argv[2]);
/* Create a socket point */
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
{
   perror("ERROR opening socket");
   exit(1);
}
server = gethostbyname(argv[1]);
if (server == NULL) {
   fprintf(stderr,"ERROR, no such host\n");
   exit(0);
}

bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
```

```
bcopy((char *)server->h_addr,
      (char *)&serv_addr.sin_addr.s_addr,
           server->h_length);
serv_addr.sin_port = htons(portno);

/* Now connect to the server */
if (connect(sockfd,&serv_addr,sizeof(serv_addr)) < 0)
{
    perror("ERROR connecting");
    exit(1);
}
/* Now ask for a message from the user, this message
* will be read by server
*/


bzero (buffer,256);
scanf ("%s", buffer);
n=write (sockfd, buffer, strlen(buffer));
bzero (buffer,256);
n=read (sockfd, buffer,255);
printf ("%s\n",buffer);


if (n < 0)
```

```
{
    perror("ERROR writing to socket");
    exit(1);
}
/* Now read server response */
bzero(buffer,256);
n = read(sockfd,buffer,255);
if (n < 0)
{
    perror("ERROR reading from socket");
    exit(1);
}
printf("%s\n",buffer);
bzero(buffer,256);
scanf ("%s", buffer);
n=write (sockfd, buffer, strlen(buffer));
//start from croup
buffer[0]='2';
while (strcmp(buffer,"1")!=0) {
     bzero(buffer,256);
     n = read(sockfd,buffer,255);
     if (strcmp(buffer,"1"))
            printf("%s\n",buffer);
}
bzero(buffer,256);
```

```
        scanf ("%s", buffer);
        n=write (sockfd, buffer, strlen(buffer));
        bzero(buffer,256);
    n = read(sockfd,buffer,255);
    printf("%s\n",buffer);
    return 0;
}
```

TCP_Server_win

```
/*
    Bind socket to port 8888 on localhost
*/
#include<io.h>
#include<stdio.h>
#include<winsock2.h>
#include <time.h>
#include <windows.h>
DWORD Thread1, Thread2;
#pragma comment(lib,"ws2_32.lib") //Winsock Library
DWORD WINAPI thread2(LPVOID lpParameter) {
       int recv_size;
       char buffer[256];
       char buffer_file[256];
       int N = 256;
       int n;
       int c;
       int i = 0;
       int player;
       int number, odd;
       char my_number [3];
       char my_odd_or_even;
       char my_bet [5];
        SOCKET sock = *((SOCKET*)lpParameter);

//     n = read(sock, buffer, N - 1);

       FILE * fo;

       fo = fopen("test.txt", "r");
       fgets(buffer_file, 50, fo);
       fclose(fo);

       memset(buffer, 0, sizeof (buffer));
       recv_size=recv(sock , buffer , 2000 , 0);
        buffer[recv_size] = '\0';
       printf("Here is the message: %c\n", buffer);
       if (buffer_file[2] == '0') {
              if (buffer[2] == '1') {
                     fo = fopen("test.txt", "a+");
                     fputs("\n", fo);
                     fputs(buffer, fo);
```

```
                        fclose(fo);
                        player = 1;
                //      write(sock, "You are in game, player\n", 24);
                        send(sock , "You are in game, player\n" , 24 , 0);

                } else {
                //      write(sock, "u cant be a croup\n", 18);
                send(sock , "u cant be a croup\n" ,18 , 0);

                }
        } else {
                if (buffer[2] == '0') {
                        fo = fopen("test.txt", "a+");

                        fputs(buffer, fo);
                        fclose(fo);
                        player = 0;
                //      write(sock, "You are in game, croup\n", 23);
                        send(sock ,"You are in game, croup\n" ,23 , 0);

                }
                else {
                        //write(sock, "u cant be a player", 18);
                        send(sock ,"u cant be a player\n" ,18 , 0);

                 }

        }
        memset(buffer, 0, sizeof (buffer));
        if (player == 0) {
                //write(sock, "waiting...players are making bets\n", 34);
                        send(sock ,"waiting...players are making bets\n" ,34 , 0);

        }
        else
        //      write(sock, "make your bet", 13);
        send(sock ,"make your bet" ,13 , 0);

//      n = read(sock, buffer, N - 1);
recv_size=    recv(sock , buffer , 2000 , 0);
 buffer[recv_size] = '\0';
        printf("Here is the message: buffer[0]= %c, player= %d\n", buffer[0],
                        player);
        if (player == 0) {
                if (buffer[0] == '1')
                        printf("its time to show bets\n");
                fo = fopen("bets.txt", "r");
                while (fgets(buffer, 50, fo) != NULL) {
        //              printf("bets: %s\n", buffer);
                //      write(sock, buffer, strlen(buffer));
                        send(sock ,buffer ,strlen(buffer) , 0);
                }
        } else {
        //      write(sock, "i got your bet", 14);
                send(sock ,"i got your bet", 14 , 0);
                fo = fopen("bets.txt", "a+");
                fputs(buffer, fo);
                fputs("\n", fo);
                fclose(fo);
                while (buffer[i]!=';') {
                        my_number[i]=buffer[i];
                                i++;
                }
                i++;
                my_odd_or_even=buffer[i];
```

```
                i++;
                int j=0;
                for(i=5;i <10;i++,j++) {
                        my_bet[j]=buffer[i];

                }

        }

        if (player == 0) {
                memset(buffer, 0, sizeof (buffer));
//              n = read(sock, buffer, N - 1);
                recv_size=recv(sock , buffer , 2000 , 0);
                 buffer[recv_size] = '\0';
                if (buffer[0] == 's') {
                        srand(time(NULL));
                        number = rand() % 100;
                        odd = rand() % 2;
                        fo = fopen("game.txt", "a+");
                        fprintf(fo, "%d", number);
                        fputs("\n", fo);
                        fprintf(fo, "%d", odd);
                        fclose(fo);
                }

        }
        int win1=5;
        int win2=5;
        if (player == 1) {
//              n = read(sock, buffer, N - 1);
        recv_size=    recv(sock , buffer , 2000 , 0);
         buffer[recv_size] = '\0';
                if (buffer[0] == 'r') {
                        printf("new check\n");

                        fo = fopen("game.txt", "r");
                        fgets(buffer, 50, fo);

                        if (strcmp(buffer,my_number)==0)
                                win1=1;

                        else
                                win1=0;
                        fgets(buffer, 50, fo);
                        if (buffer[0]==my_odd_or_even)
                                                win2=1;

                                        else
                                                win2=0;

                        fclose(fo);
                        if (win1==1||win2==1)
                        //      write(sock, "Granz, you won", 14);
                                        send(sock ,"Gratz, you won", 14 , 0);
                        else
                        //      write(sock, "you loose", 9);
                                send(sock ,"you loose", 9 , 0);
                }
        }

}
int main(int argc , char *argv[])
{
    WSADATA wsa;
    SOCKET s , new_socket;
```

```
    struct sockaddr_in server , client;
    int c;
    char *message;
       char buffer [256];
    printf("\nInitialising Winsock...");
    if (WSAStartup(MAKEWORD(2,2),&wsa) != 0)
    {
        printf("Failed. Error Code : %d",WSAGetLastError());
        return 1;
    }

    printf("Initialised.\n");

    //Create a socket
    if((s = socket(AF_INET , SOCK_STREAM , 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( 8888 );

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

    puts("Bind done");

    //Listen to incoming connections
    listen(s , 3);

    //Accept and incoming connection
    puts("Waiting for incoming connections...");
        c = sizeof(struct sockaddr_in);
      while (1) {


    new_socket = accept(s , (struct sockaddr *)&client, &c);
    if (new_socket == INVALID_SOCKET)
    {
        printf("accept failed with error code : %d" , WSAGetLastError());
    }

    puts("Connection accepted");
           DWORD myThreadID;

               HANDLE myHandle1 = CreateThread(0, 0, thread2,&new_socket, 0, &myThreadID);
        CloseHandle(myHandle1);
         }
    closesocket(s);
    WSACleanup();

    return 0;
}
```

Tcp_client_win

```c
/*
    Create a TCP socket
*/

#include<stdio.h>
#include<winsock2.h>

#pragma comment(lib,"ws2_32.lib") //Winsock Library

int main(int argc , char *argv[])
{
    WSADATA wsa;
    SOCKET s;
    struct sockaddr_in server;
    char *message , server_reply[2000];
    int recv_size;
       char buffer[256];
    printf("\nInitialising Winsock...");
    if (WSAStartup(MAKEWORD(2,2),&wsa) != 0)
    {
        printf("Failed. Error Code : %d",WSAGetLastError());
        return 1;
    }

    printf("Initialised.\n");

    //Create a socket
    if((s = socket(AF_INET , SOCK_STREAM , 0 )) == INVALID_SOCKET)
    {
        printf("Could not create socket : %d" , WSAGetLastError());
    }

    printf("Socket created.\n");


    server.sin_addr.s_addr = inet_addr("127.0.0.1");
    server.sin_family = AF_INET;
    server.sin_port = htons(8888);

       //Connect to remote server
    if (connect(s , (struct sockaddr *)&server , sizeof(server)) < 0)
    {
        puts("connect error");
        return 1;
    }

    puts("Connected");

    //Send some data

       memset(buffer, 0, sizeof (buffer));
    scanf ("%s", buffer);

  //  n=write (sockfd, buffer, strlen(buffer));
       send(s ,buffer ,strlen(buffer) , 0);
       getchar ();
    memset(buffer, 0, sizeof (buffer));
 //   n=read (sockfd, buffer,255);
       recv_size=recv(s , buffer , 2000 , 0);
       buffer[recv_size] = '\0';
    printf ("%s\n",buffer);
```

```
    /* Now read server response */
    memset(buffer, 0, sizeof (buffer));
 //   n = read(sockfd,buffer,255);
recv_size=    recv(s , buffer , 2000 , 0);
  buffer[recv_size] = '\0';
    printf("%s\n",buffer);
     memset(buffer, 0, sizeof (buffer));
    scanf ("%s", buffer);
  //  n=write (sockfd, buffer, strlen(buffer));
       send(s ,buffer ,strlen(buffer) , 0);
       getchar ();
    //start from croup


      memset(buffer, 0, sizeof (buffer));
 //   n = read(sockfd,buffer,255);
       recv_size=  recv(s , buffer , 2000 , 0);
         buffer[recv_size] = '\0';
    printf("%s\n",buffer);
    memset(buffer, 0, sizeof (buffer));
        scanf ("%s", buffer);
        send(s ,buffer ,strlen(buffer) , 0);
              getchar ();
      memset(buffer, 0, sizeof (buffer));
   recv_size= recv(s , buffer , 2000 , 0);
     buffer[recv_size] = '\0';
    printf("%s\n",buffer);
    return 0;
}
```

Udp_server_win

```
// udp_client.cpp: определяет точку входа для консольного приложения.
//



#include "stdafx.h"
#include <io.h>
#include <stdio.h>
#include <winsock2.h>
#include <stdlib.h>
#include <string.h>
#include <direct.h>
#include <locale.h>
#include <sys/types.h>
#include <limits.h>
#include <sys/stat.h>
#include <time.h>
#pragma comment(lib, "WS2_32.lib")

//struct sockaddr_in servaddr,
//cliaddr;
void thread2(int sock) {
       char buffer[256];
       int N = 256;
       int n;
       char mesg[256];
```

```
        char buffer_file[256];
         struct sockaddr_in cliaddr,si_other;
        //     int n;
        int c;
        int i = 0;
        int player;
        int number, odd;
        char my_number[3];
        char my_odd_or_even;
        char my_bet[5];
        memset(buffer,'\0',N);
        int len;
        len = sizeof(cliaddr);



        //n = recvfrom(sock, buffer, N, 0, (struct sockaddr *) &cliaddr, &len);
//      buffer[n]='\0';
//      printf("Here is the message: %s\n", buffer);

//      sendto(sock, "hi", 2, 0, (struct sockaddr*) &cliaddr, len);


          /*   n = recvfrom(sock,buffer,N,0,(struct sockaddr *)&cliaddr,&len);
         sendto(sock,buffer,N,0,(struct sockaddr *)&cliaddr,sizeof(cliaddr));
         printf("--------------------------------------------------------\n");
         //  buffer[n] = 0;
         printf("Received the following:\n");
         printf("%s",buffer);
         printf("--------------------------------------------------------\n");*/
        n = recvfrom(sock, buffer, N, 0, (struct sockaddr *) &cliaddr, &len);
//      buffer[n]='\0';
        printf("Here is the message: %s\n", buffer);
        FILE * fo;
        fo = fopen("test.txt", "r");

        fgets(buffer_file, 50, fo);
        //memset(buffer, 0, sizeof (buffer));
        fclose(fo);
        printf("Here is the message: %s\n", buffer_file);
        if (buffer_file[2] == '0') {
                if (buffer[2] == '1') {
                        fo = fopen("test.txt", "a+");
                        fputs("\n", fo);
                        fputs(buffer, fo);
                        fclose(fo);
                        player = 1;
                        //     write(sock, "You are in game, player\n", 24);
                        sendto(sock, "You are in game, player\n",23, 0,(struct sockaddr*)
&cliaddr, len);
                } else
                        //            write(sock, "u cant be a croup\n", 18);
                        sendto(sock, "u cant be a croup\n", 17, 0,(struct sockaddr*)
&cliaddr, len);
        } else {
                if (buffer[2] == '0') {
                        fo = fopen("test.txt", "a+");

                        fputs(buffer, fo);
                        fclose(fo);
                        player = 0;
                        //write(sock, "You are in game, croup\n", 23);
                        sendto(sock, "You are in game, croup\n", N, 0,(struct sockaddr*)
&cliaddr, len);
```

```
		} else {
			//	write(sock, "u cant be a player", 18);
			sendto(sock, "u cant be a player", N, 0,(struct sockaddr*) &cliaddr,
len);
		}

	}
	memset(buffer,'\0',N);
	if (player == 0)
		sendto(sock, "waiting...players are making bets\n", N, 0,
			(struct sockaddr *) &cliaddr, sizeof(cliaddr));
	else
		sendto(sock, "make your bet", N, 0, (struct sockaddr *) &cliaddr,
			sizeof(cliaddr));
	//n = read(sock, buffer, N - 1);
	n = recvfrom(sock, buffer, N, 0, (struct sockaddr *) &cliaddr, &len);
	printf("Here is the message: buffer[0]= %c, player= %d\n", buffer[0],
			player);
	if (player == 0) {
		if (buffer[0] == '1')
			printf("its time to show bets\n");
		fo = fopen("bets.txt", "r");
		while (fgets(buffer, 50, fo) != NULL) {
			//		printf("bets: %s\n", buffer);
			//	write(sock, buffer, strlen(buffer));
			sendto(sock, buffer, N, 0, (struct sockaddr *) &cliaddr,
				sizeof(cliaddr));
		}
	} else {
		//write(sock, "i got your bet", 14);
		sendto(sock, "i got your bet", N, 0, (struct sockaddr *) &cliaddr,
			sizeof(cliaddr));
		fo = fopen("bets.txt", "a+");
		fputs(buffer, fo);
		fputs("\n", fo);
		fclose(fo);
		while (buffer[i] != ';') {
			my_number[i] = buffer[i];
			i++;
		}
		i++;
		my_odd_or_even = buffer[i];
		i++;
		int j = 0;
		for (i = 5; i < 10; i++, j++) {
			my_bet[j] = buffer[i];

		}

	}

	if (player == 0) {
		memset(buffer,'\0',N);
		//n = read(sock, buffer, N - 1);
		n = recvfrom(sock, buffer, N, 0, (struct sockaddr *) &cliaddr, &len);
		if (buffer[0] == 's') {
			srand(time(NULL));
			number = rand() % 100;
			odd = rand() % 2;
			fo = fopen("game.txt", "a+");
			fprintf(fo, "%d", number);
			fputs("\n", fo);
			fprintf(fo, "%d", odd);
			fclose(fo);
		}
```

```
		}
		int win1 = 5;
		int win2 = 5;
		if (player == 1) {
			//n = read(sock, buffer, N - 1);
			n = recvfrom(sock, buffer, N, 0, (struct sockaddr *) &cliaddr, &len);
			if (buffer[0] == 'r') {
				printf("new check\n");

				fo = fopen("game.txt", "r");
				fgets(buffer, 50, fo);

				if (strcmp(buffer, my_number) == 0)
					win1 = 1;

				else
					win1 = 0;
				fgets(buffer, 50, fo);
				if (buffer[0] == my_odd_or_even)
					win2 = 1;

				else
					win2 = 0;

				fclose(fo);
				if (win1 == 1 || win2 == 1)
					//	write(sock, "Granz, you won", 14);
					sendto(sock, "Granz, you won", N, 0,
							(struct sockaddr *) &cliaddr, sizeof(cliaddr));
				else
					//	write(sock, "you loose", 9);
					sendto(sock, "you loose", N, 0, (struct sockaddr *) &cliaddr,
							sizeof(cliaddr));
			}
		}
}

#define BUFLEN 512  //Max length of buffer
#define PORT 8888   //The port on which to listen for incoming data

int main()
{
    SOCKET s;
    struct sockaddr_in server, si_other;
    int slen , recv_len;
    char buf[BUFLEN];
    WSADATA wsa;

    slen = sizeof(si_other) ;

    //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");

    //keep listening for data
 thread2(s);


    closesocket(s);
    WSACleanup();

    return 0;
}
```

Udp_client_win

```
/*
    Create a TCP socket
*/

#include<stdio.h>
#include<winsock2.h>

#pragma comment(lib,"ws2_32.lib") //Winsock Library

int main(int argc , char *argv[])
{
    WSADATA wsa;
    SOCKET s;
    struct sockaddr_in server;
    char *message , server_reply[2000];
    int recv_size;
       char buffer[256];
    printf("\nInitialising Winsock...");
    if (WSAStartup(MAKEWORD(2,2),&wsa) != 0)
    {
        printf("Failed. Error Code : %d",WSAGetLastError());
        return 1;
    }

    printf("Initialised.\n");

    //Create a socket
    if((s = socket(AF_INET , SOCK_STREAM , 0 )) == INVALID_SOCKET)
    {
        printf("Could not create socket : %d" , WSAGetLastError());
    }

    printf("Socket created.\n");


    server.sin_addr.s_addr = inet_addr("127.0.0.1");
```

```
    server.sin_family = AF_INET;
    server.sin_port = htons(8888);

       //Connect to remote server
    if (connect(s , (struct sockaddr *)&server , sizeof(server)) < 0)
    {
        puts("connect error");
        return 1;
    }

    puts("Connected");

    //Send some data

       memset(buffer, 0, sizeof (buffer));
    scanf ("%s", buffer);

  //  n=write (sockfd, buffer, strlen(buffer));
       send(s ,buffer ,strlen(buffer) , 0);
       getchar ();
    memset(buffer, 0, sizeof (buffer));
 //   n=read (sockfd, buffer,255);
       recv_size=recv(s , buffer , 2000 , 0);
       buffer[recv_size] = '\0';
    printf ("%s\n",buffer);



    /* Now read server response */
    memset(buffer, 0, sizeof (buffer));
 //   n = read(sockfd,buffer,255);
recv_size=    recv(s , buffer , 2000 , 0);
  buffer[recv_size] = '\0';
    printf("%s\n",buffer);
     memset(buffer, 0, sizeof (buffer));
    scanf ("%s", buffer);
  //  n=write (sockfd, buffer, strlen(buffer));
       send(s ,buffer ,strlen(buffer) , 0);
       getchar ();
    //start from croup


      memset(buffer, 0, sizeof (buffer));
 //   n = read(sockfd,buffer,255);
       recv_size=  recv(s , buffer , 2000 , 0);
         buffer[recv_size] = '\0';
    printf("%s\n",buffer);
    memset(buffer, 0, sizeof (buffer));
        scanf ("%s", buffer);
        send(s ,buffer ,strlen(buffer) , 0);
              getchar ();
      memset(buffer, 0, sizeof (buffer));
   recv_size= recv(s , buffer , 2000 , 0);
     buffer[recv_size] = '\0';
    printf("%s\n",buffer);
    return 0;
}
```