Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Кафедра информационных технологий и систем

**Сетевая игра «Палач»**

Курсовая работа

по дисциплине:

«Операционные системы»

Выполнил:

Студент группы 9091

Прокофьев М.Ю.

«\_\_»\_\_\_\_\_.2021г.

Проверил:

Ананьев В.В.

«\_\_»\_\_\_\_\_.2021г.

Великий Новгород  
2021

**ОГЛАВЛЕНИЕ**

ВВЕДЕНИЕ....................................................................................................................................

ГЛАВА 1. РАЗРАБОТКА ПРОЕКТА............................................................................................

1.1 Постановка задачи...................................................................................................................

1.2 Выбор инструментальных средств........................................................................................

1.3 Выбор модели..........................................................................................................................

1.4 Алгоритм решения задачи.......................................................................................................

ГЛАВА 2. РЕАЛИЗАЦИЯ ПРОЕКТА.........................................................................................

2.1 Структура клиента и сервера.................................................................................................

2.2 Программный интерфейс сервера.........................................................................................

2.3 Описание программной реализации клиента.......................................................................

2.4 Системные вызовы..................................................................................................................

2.5 Пример работы приложения...................................................................................................

2.6 Результат работы.....................................................................................................................

ЗАКЛЮЧЕНИЕ..............................................................................................................................

Код из файла server.сpp........................................................................................ ..…………….

Код из файла client.сpp....................................................................................................................

**ВВЕДЕНИЕ**

Цель данной работы – создание простейшей консольной версии игры «Палач», а также получение практических навыков разработки и реализации многопользовательских сетевых приложений под управлением операционной системы семейства Unix/Linux.

**Палач**— игра для двух людей, связанная со словами и буквами. Особенность проектируемой игры заключается в том, что она должна работать под операционной системой Linux.

Правила игры:

В игре участвуют только два игрока.

Один игрок (водящий) загадывает слово и рисует на листе такое количество подчёркиваний, сколько букв в слове. Другой игрок начинает называть буквы, чтобы отгадать слово. Если буква есть в слове, то водящий обязан вписать её на своё место в слово (если таких букв несколько — то вписываются все). Второму игроку даётся 6 попыток, чтобы полностью угадать слово( по количеству частей тела человека). Однако попытка не сгорает, если за свой ход игрок угадает хотя бы одну букву из загаданного слова. За один ход игрок может назвать только одну букву. Если второй игрок успевает отгадать слово за 6 попыток, тогда он побеждает, а первый проигрывает, в противном случае, побеждает первый игрок, а второй проигрывает. Затем идёт смена, то есть игроки меняются ролями. (Второй становится водящим, а первый должен угадать слово.)

Функции, реализованные в игре:

1. Подключение двух игроков
2. Возможность для ведущего вводить слово, а для игрока вводить буквы
3. Консольный интерфейс для ввода слова, букв и вывода результата

**ГЛАВА 1. РАЗРАБОТКА ПРОЕКТА**

**1.1 Постановка задачи**

Задачей данной курсовой работы является разработка сетевой игры «Виселица», которая рассчитана на двух игроков. Игра должна работать под управлением операционной системы семейства Unix/Linux. Сетевая часть игры должна быть выполнена на языке C++ с помощью сокетов предоставляемых системной библиотекой <sys/socket.h>.

**1.2 Выбор инструментальных средств**

Для реализации сетевого взаимодействия используются протокол передачи данных: TCP.

Протокол TCP предоставляет транспортные услуги, отличающиеся от услуг UDP. Вместо ненадежной доставки датаграмм без установления соединений, он обеспечивает гарантированную доставку с установлением соединений между прикладными процессами в виде байтовых потоков.

Протокол TCP используется в тех случаях, когда требуется надежная доставка сообщений. Он освобождает прикладные процессы от необходимости использовать таймауты и повторные передачи для обеспечения надежности.

**1.3 Выбор модели**

Для взаимодействия между игроками выбрана клиент-серверная модель. Сервер обеспечивает хранение всех данных игры.

**1.4 Алгоритм решения задачи**

Подключение игрока/игроков происходит следующим образом:

К серверу подключаются первый и второй игрок, введя адрес сервера и номер порта.

Ход игры:

Сервер ожидает ввода слова и записывает его в массив. Игрок по буквам угадывает слово, до тех пор, пока слово не будет угадано или не закончиться количество попыток. Игроку выводятся промежуточные результаты. После окончания игроки меняются местами.

**ГЛАВА 2. РЕАЛИЗАЦИЯ ПРОЕКТА**

**2.1 Структура клиента и сервера**

Соединение между клиентом и сервером. Для игры также используются некоторые дополнительные функции.

**2.2 Программный интерфейс сервера**

Сервер получает от клиента длину слова, а затем само слово этой длины. Также отсылает длину слова другому игроку. В случае победы, клиент отправляет на сервер '\0' символ. И сервер сообщает второму игроку о победе, отправляя код 0. В случае проигрыша для игрока код — 9999, для ведущего — 1. В случае успешного хода, сервер отправляет количество найденных букв и затем координаты этих букв.

**2.3 Описание программной реализации клиента**

При запуске клиента происходит подключение к серверу. У клиента два разных состояния: когда он ходит и когда он загадывает слово. Коды о его состоянии он получает от сервера.

**2.4 Системные вызовы**

**1) socket (int domain, int type, int protocol) -** cлужит для создания сокета в ОС. Имеет три парметра. Первый указывает, к какому семейству протоколов относится создаваемый сокет, а второй и третий параметры определяют конкретный протокол внутри данного семейства.

**2) connect(int *sockfd*, const struct sockaddr \**serv\_addr*, socklen\_t *addrlen*) -** служит для установки логического соединения со стороны клиента. Имеет три параметра: дескриптор активного сокета, через который будет устанавливаться соединение, полный адрес сокета сервера и его длина.

**3) inet\_pton(int** *af***, const char \****src***, void \****dst***) -**  преобразует строку символов *src* в сетевой адрес (типа *af*), затем копирует полученную структуру с адресом в *dst*. На текущий момент поддерживаются следующие типы адресов: **AF\_INET и AF\_INET6.** Если сетевой адрес был успешно преобразован, то возвращается положительное значение.

**4) read() и write()** - используются для чтения и записи файлов. Могут быть использованы для обмена данных, если в качестве параметров вместо дескрипторов файлов в них задаются дескрипторы сокетов.

**5) bind(int***sockfd***, struct sockaddr** \**my\_addr***, socklen\_t***addrlen***) -** настраивает адрес для созданного сокета. Первый параметр вызова должен содержать дескриптор сокета, для которого производится настройка адреса. Второй и третий параметры задают этот адрес.

**6) listen(int *s*, int *backlog*) -** прослушивает соединения на сокете. Позволяет показать готовность принимать соединения и задать лимит входящих соединений. В качестве первого параметра используется дескриптор сокета. Второй параметр определяет максимальную длину очереди входящих соединений.

**7) htons(uint16\_t** *hostshort***) -** преобразует узловой порядок расположения байтов положительного короткого целого *hostshort* в сетевой порядок расположения байтов.

**8) ntohs(uint16\_t** *netshort***) -** преобразует сетевой порядок расположения байтов положительного короткого целого *netshort* в узловой порядок расположения байтов.

**9) accept(int***s***, struct sockaddr** \**addr***, socklen\_t** \**addrlen***) –** эта функция используется сервером для принятия связи на сокет. Сокет должен быть уже слушающим в момент вызова функции. Если сервер устанавливает связь с клиентом, то функция accept возвращает новый сокет-дескриптор, через который и происходит общение клиента с сервером. Пока устанавливается связь клиента с сервером, функция accept блокирует другие запросы связи с данным сервером, а после установления связи "прослушивание" запросов возобновляется.

**2.5 Пример работы приложения**

Сначала надо запустить программу сервер, введя в командой строке:

./server [port].

Затем надо запустить программу клиент, для которой формат ввода данных имеет следующий вид:

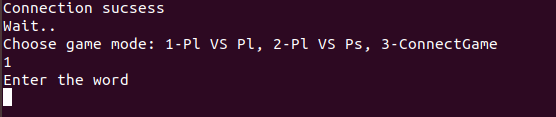
./client [ip] [port]

**Режим игрок против игрока**.

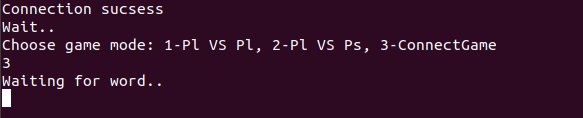
Для этого режима программа клиент должна быть запущена два раза.

Игрок, который будет загадывать слово, вводит 1 и ждёт подключения второго игрока. Другой игрок вводит значение 3 для подключения к водящему.

Результат, выведенный на экран 1 игрока:



Результат, выведенный на экран 2 игрока:

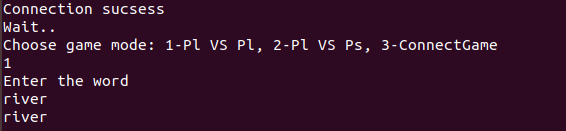


Теперь первый игрок должен ввести слово, которое он хочет загадать. Другой игрок в это время ждёт, пока водящий не загадает слово.

После того как слово загадано, само слово и его длина будет хранится на сервере.

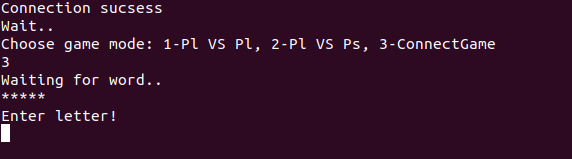
Результат, выведенный на экран программы сервер:

C:\Users\user\Desktop\Безымянный.png

Результат, выведенный на экран 1 игрока: 

(Если загаданное слово снова вывелось на экран, значит слово отправлено.)

Результат, выведенный на экран 2 игрока:

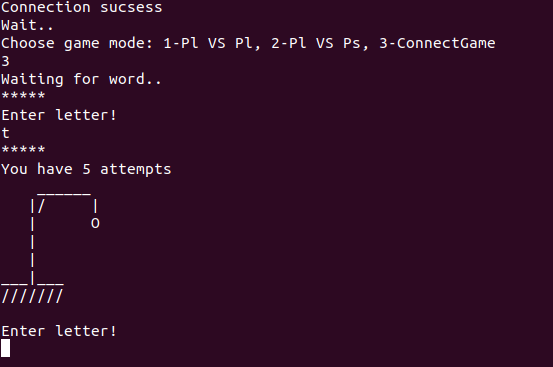


Теперь водящий будет ждать результата от второго игрока (угадал 2 игрок слово или нет).

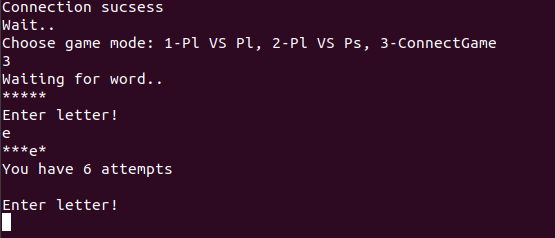
Ему доступно 6 попыток, чтобы угадать слово. За раз надо вводить 1 букву.

Попытка не тратится, если угадана хотя бы одна буква.

В случае неудачи, результат, выведенный на экран 2 игрока:



В случае успеха, результат, выведенный на экран 2 игрока:



Смена игроков происходит по окончанию действий игрока, который пытается угадать слово.

Победа второго игрока

Результат, выведенный на экран 1 игрока:



Результат, выведенный на экран 2 игрока:

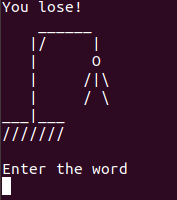


Победа первого игрока

Результат, выведенный на экран 1 игрока:



Результат, выведенный на экран 2 игрока:



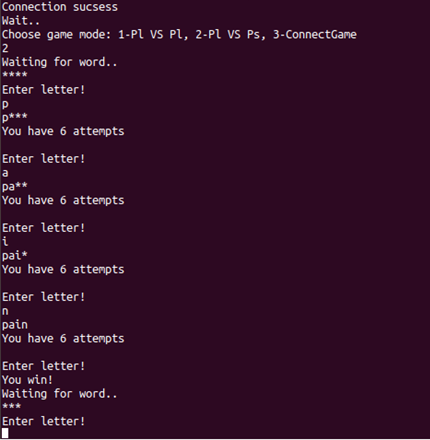
**Режим игрок против компьютера**

Здесь программа клиент должна быть запущена всего 1 раз.

Игрок вводит значение 2 для выбора режима игры против компьютера. В этом режиме компьютер и игрок не будут меняться ролями, компьютер всегда будет водящим, то есть будет произвольно загадывать слова из заранее записанного массива слов, а игрок должен будет их угадать.

Пример:

Клиент



Сервер



**2.6 Результат работы**

В ходе выполнения данной курсовой работы была создана работающая сетевая игра «Виселица», поддерживающая операционной системы семейства Unix/Linux, в которой для сетевого взаимодействия игроков используются клиент-серверная модель, а обмен данными осуществляется через TCP – сокеты.

**ЗАКЛЮЧЕНИЕ**

Таким образом, результат курсовой работы соответствует требованиям к реализации. Приложение работает на операционных системах семейства Linux. Для построения сетевой части приложения использован язык C++ и компилятор g++. Сетевое взаимодействие было построено с помощью протокола TCP. В целом считаю, что справился с поставленной задачей.

Код с комментариями из файла server.сpp

#include <stdio.h>

#include <sys/shm.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/sem.h>

#include <sys/msg.h>

#include <sys/ipc.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <signal.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <errno.h>

#include <time.h>

#include <string>

#include <iostream>

int count(char\* str) {

int i=0;

while(str[i]!='\0')

{

i++;

}

return i;

}

void game(int listenfd, int cl1, int cl2) {

while (1) {

int check = 0;

check = htons (check);

write (cl1, &check, sizeof(int)); // отпраляем статус начала0

check = 1;

check = htons (check);

write (cl2, &check, sizeof(int)); // отпраляем статус начала1

char word[100];

read (cl1, &check, sizeof(int)); // получаем длинну строки

check = ntohs (check);

int lengt = check;

read (cl1, word, (lengt+1)\*sizeof(char)); //получем само слово

std::cout << "Word: " << word << '\n';

if(word[0] == '/' && word[1] == 's') {

std::cout << "Wait for new connection" << '\n';

cl1 = accept(listenfd, (struct sockaddr\*)NULL, NULL); //ждем подключеня 1 клиента

check = 0;

check = htons (check);

write (cl1, &check, sizeof(int));

read (cl1, &check, sizeof(int)); // получаем длинну строки

check = ntohs (check);

lengt = check;

read (cl1, word, (lengt+1)\*sizeof(char)); //получем само слово

std::cout << "Word: " << word << '\n';

}

printf("%d\n", lengt);

lengt = htons (lengt);

write (cl2, &lengt, sizeof(int)); // прередаем длинну слова угадывающему

lengt = ntohs (lengt);

int failAtt = 0;

while (1)

{

char leter;

read (cl2, &leter, sizeof(char)); // читаем букву

//конец игры при победе первого и свап клиентов

if (leter == '\0') {

check = 0;

check = htons(check);

write (cl1, &check, sizeof(int)); // отправляем первому сообщ о победе

check = ntohs(check);

int swap;

swap = cl1;

cl1 = cl2;

cl2 = swap;

break;

}

//конец игры при победе второго и свап клиентов

if(failAtt == 5) {

check = 9999;

check = htons(check);

write (cl2, &check, sizeof(int)); // отправляем проигрыш

check = 1;

check = htons(check);

write (cl1, &check, sizeof(int)); // отправляем первому сообщ о проирыше

check = ntohs(check);

int swap;

swap = cl1;

cl1 = cl2;

cl2 = swap;

break;

}

int numLet = 0;

for (int i=0; i<lengt; i++)

{

if (word[i]==leter) numLet++;

}

numLet = htons(numLet);

write (cl2, &numLet, sizeof(int)); // отправляем число найденных букв

if (numLet == 0) failAtt++;

for (int i=0; i<lengt; i++)

{

if (word[i]==leter)

{

i=htons(i);

write (cl2, &i, sizeof(int)); // отправляем координаты каждой буквы

i=ntohs(i);

}

}

}

}

}

char words[][100] = {"pain", "shoot", "dead", "fun", "illusion", "demention", "implementation"};

int numword = 7;

void gamePs(int listenfd, int cl1) {

while (1) {

int check = 1;

check = htons (check);

write (cl1, &check, sizeof(int)); // отпраляем статус начала0

srand((unsigned) (time(0)));

char\* word = words[rand()%numword];

int lengt = count(word);

std::cout << "Word: " << word << '\n';

printf("%d\n", lengt);

lengt = htons (lengt);

write (cl1, &lengt, sizeof(int)); // прередаем длинну слова угадывающему

lengt = ntohs (lengt);

int failAtt = 0;

while (1)

{

char leter;

read (cl1, &leter, sizeof(char)); // читаем букву

if (leter == '\0') {

break;

}

if(failAtt == 5) {

check = 9999;

check = htons(check);

write (cl1, &check, sizeof(int)); // отправляем проигрыш

break;

}

int numLet = 0;

for (int i=0; i<lengt; i++)

{

if (word[i]==leter) numLet++;

}

numLet = htons(numLet);

write (cl1, &numLet, sizeof(int)); // отправляем число найденных букв

if (numLet == 0) failAtt++; //увеличение количества ошибок

for (int i=0; i<lengt; i++)

{

if (word[i]==leter)

{

i = htons(i);

write (cl1, &i, sizeof(int)); // отправляем координаты каждой совпадающей буквы

i = ntohs(i);

}

}

}

}

}

int main(int argc, char \*argv[]) {

int listenfd = 0;

struct sockaddr\_in serv\_addr;

char sendBuff[1025];

time\_t ticks;

listenfd = socket(AF\_INET, SOCK\_STREAM, 0); //инициализация сокета

memset(&serv\_addr, '0', sizeof(serv\_addr)); //обнуление serv\_addr

memset(sendBuff, '0', sizeof(sendBuff)); //обнуление sendBuff

//настройки сокета

serv\_addr.sin\_family = AF\_INET; //IPv4

serv\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY); //ip

serv\_addr.sin\_port = htons(atoi(argv[1])); //порт

bind(listenfd, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr)); //состыковка сокета и настройки сокета

listen(listenfd, 10); //прослушивание сокета максимум на 10 клиентов

int cl1 = accept(listenfd, (struct sockaddr\*)NULL, NULL); //ждем подключеня 1 клиента

int check;

read (cl1, &check, sizeof(int)); //получение типа игры для первого клиента

check = ntohs(check);

if (check == 1) {

int cl2 = accept(listenfd, (struct sockaddr\*)NULL, NULL); //ждем подключеня 2 клиента

game(listenfd, cl1, cl2);

}

else

gamePs(listenfd, cl1);

}

Код из файла client.сpp

#include <stdio.h>

#include <sys/shm.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/sem.h>

#include <sys/msg.h>

#include <sys/ipc.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <signal.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <errno.h>

#include <string>

#include <iostream>

//функция получения размера строки

int count(char\* str) {

int i=0;

while(str[i]!='\0')

{

i++;

}

return i;

}

void game(int sockfd) {

while(1)

{

int check;

read (sockfd, &check, sizeof(int)); //чтение типа игры из сокета

check = ntohs(check); //преобразование типа игры в нормальный вид

if(check == 0) //ход наш (загадываем слово)

{

char word[100];

std::cout << "Enter the word" << '\n';

std::cin >> word;

std::cout << word << '\n';

int num = count(word);

num = htons(num);

write (sockfd, &num, sizeof(int)); // передаем число букв в слове

num = ntohs(num);

write (sockfd, word, (num+1)\*sizeof(char)); // передаем само слово

if (word[0] == '/' && word[1] == 's')

break;

read (sockfd, &check, sizeof(int)); // получаем итог - 1 - пр 2-по

check = ntohs(check);

if (check == 1) {

std::cout << "Your rival is lose " << '\n';

}

else std::cout << "Your rival is win!"<< '\n';

}

else //ход не наш (угадываем слово)

{

int trying = 6; //количество попыток

printf("Waiting for word..\n");

int numLet;

read (sockfd, &numLet, sizeof(int)); //получаем длинну слова

numLet = ntohs(numLet);

char word[100];

for (int i=0; i<numLet; i++) //Инициализируем игровой массив

{

word[i]='\*';

}

word[numLet]='\0';

std::cout << word << '\n';

bool fail = false;

do {

std::cout << "Enter letter!\n";

char leter;

int win = 0;

//проверка на победу

for (int i=0; i<numLet; i++)

{

if (word[i]=='\*') break;

else win++;

}

if (win == (numLet))

{

leter='\0';

write (sockfd, &leter, sizeof(char));

std::cout<<"You win!" << '\n';

break;

}

std::cin >> leter;

write (sockfd, &leter, sizeof(char)); // отправляем букву

read (sockfd, &check, sizeof(int)); // читаем число таких букв

check = ntohs(check);

if (check == 0) {

trying--;

}

if (check == 9999) {

fail = true;

std::cout << "You lose!" << '\n';

std::cout << " \_\_\_\_\_\_ " << '\n';

std::cout << " |/ | " << '\n';

std::cout << " | O " << '\n';

std::cout << " | /|\\ " << '\n';

std::cout << " | / \\ " << '\n';

std::cout << "\_\_\_|\_\_\_ " << '\n';

std::cout << "/////// " << '\n';

std::cout << '\n';

break;

}

for (int i=0; i<check; i++)

{

int pos;

read (sockfd, &pos, sizeof(int)); // читаем координаты буквы

pos = ntohs(pos);

word[pos] = leter;

}

std::cout << word << '\n';

if (trying > 1)

{

std::cout << "You have " << trying << " attempts" << '\n';

if (trying == 5)

{

std::cout << " \_\_\_\_\_\_ " << '\n';

std::cout << " |/ | " << '\n';

std::cout << " | O " << '\n';

std::cout << " | " << '\n';

std::cout << " | " << '\n';

std::cout << "\_\_\_|\_\_\_ " << '\n';

std::cout << "/////// " << '\n';

}

if (trying == 4)

{

std::cout << " \_\_\_\_\_\_ " << '\n';

std::cout << " |/ | " << '\n';

std::cout << " | O " << '\n';

std::cout << " | | " << '\n';

std::cout << " | " << '\n';

std::cout << "\_\_\_|\_\_\_ " << '\n';

std::cout << "/////// " << '\n';

}

if (trying == 3)

{

std::cout << " \_\_\_\_\_\_ " << '\n';

std::cout << " |/ | " << '\n';

std::cout << " | O " << '\n';

std::cout << " | | " << '\n';

std::cout << " | / " << '\n';

std::cout << "\_\_\_|\_\_\_ " << '\n';

std::cout << "/////// " << '\n';

}

if (trying == 2)

{

std::cout << " \_\_\_\_\_\_ " << '\n';

std::cout << " |/ | " << '\n';

std::cout << " | O " << '\n';

std::cout << " | | " << '\n';

std::cout << " | / \\ " << '\n';

std::cout << "\_\_\_|\_\_\_ " << '\n';

std::cout << "/////// " << '\n';

}

std::cout << '\n';

}

else

{

std::cout << "You have " << trying << " attempt" << '\n';

if (trying == 1)

{

std::cout << " \_\_\_\_\_\_ " << '\n';

std::cout << " |/ | " << '\n';

std::cout << " | O " << '\n';

std::cout << " | |\\ " << '\n';

std::cout << " | / \\ " << '\n';

std::cout << "\_\_\_|\_\_\_ " << '\n';

std::cout << "/////// " << '\n';

}

std::cout << '\n';

}

} while (1);

}

}

}

int main(int argc, char \*argv[])

{

int sockfd = 0, n = 0;

char recvBuff[1024];

struct sockaddr\_in serv\_addr;

if(argc != 3) { //если количество аргументов != 3, тогда завершение программы

printf("\n Usage: %s <ip of server> \n",argv[0]);

return 1;

}

memset(recvBuff, '0',sizeof(recvBuff)); //заполнение буфера нулями

if((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) //создание сокета

{

printf("\n Error : Could not create socket \n");

return 1;

}

memset(&serv\_addr, '0', sizeof(serv\_addr)); //заполнение serv\_addr нулями

serv\_addr.sin\_family = AF\_INET; //для работы с IPv4

serv\_addr.sin\_port = htons(atoi(argv[2])); //значение порта

if(inet\_pton(AF\_INET, argv[1], &serv\_addr.sin\_addr)<=0) //при ошибке подключения к серверу по ip

{

printf("\n inet\_pton error occured\n");

return 1;

}

if( connect(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0) //при ошибке подключения через соккет

{

printf("\n Error : Connect Failed \n");

return 1;

}

printf("Connection sucsess\n");

printf("Wait..\n");

//выбор типа игры

printf("Choose game mode: 1-Pl VS Pl, 2-Pl VS Ps, 3-ConnectGame\n");

int check;

do {

std::cin >> check;

} while (check != 1 && check != 2 && check != 3);

if (check !=3) {

check = htons(check); //преобразование значаения для сокета

write (sockfd, &check, sizeof(int)); //запись значения в сокет

}

game(sockfd); //запуск игры

}