Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Курсовой проект по курсу**

**«Операционные системы»**

Группа: М80-206Б-22

Студент: Кочев Д.О.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 29.12.2023

Москва, 2023

**Постановка задачи**

**Вариант 9.**

Необходимо написать консоль-серверную игру. Необходимо написать 2 программы: сервер и клиент. Сначала запускается сервер, а далее клиенты соединяются с сервером. Сервер координирует клиентов между собой. При запуске клиента игрок может выбрать одно из следующих действий (возможно больше, если предусмотрено вариантом):

* Создать игру, введя ее имя
* Присоединиться к одной из существующих игр по имени игры

«Быки и коровы» (угадывать необходимо слова). Общение между сервером и клиентом необходимо организовать при помощи очередей сообщений (например, ZeroMQ). При создании каждой игры необходимо указывать количество игроков, которые будут участвовать. То есть угадывать могут несколько игроков. Если кто-то из игроков вышел из игры, то игра должна быть продолжена.

**Общий метод и алгоритм решения**

Программа состоит из нескольких компонентов, а именно клиента, сервера, игры, и отдельно созданного словаря. Клиент - независимая программа, которую может запустить любой пользователь. Сначала идет инициализация пользователя, а именно ввод его имени и проверка того, что на сервере больше таких имен нет. Далее пользователю предлагается либо создать игру, либо подключиться уже к существующей. При создании новой игры пользователь вводит ее название и создается дочерний процесс от сервера, называемый game. В нем генерируется слово, а так же происходят проверки попыток пользователей угадать слово. При подключении к игре в словарь вносятся данные о том, что пользователь состоит в определенной игре. Программы общаются с помощью брокера сообщений, в моем случае ZMQ. После попытки пользователем угадать слово, ход передается другому, и так по кругу. Если пользователь угадал слово - игра завершается, после чего можно опять создать игру или подключиться к другой.

**Код программы**

**server.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <fcntl.h>

#include <sys/wait.h>

#include <mqueue.h>

#include <zmq.h> // gcc server.c -o server -lzmq

#include "my\_dict.h"

#define SERVER\_PORT "tcp://127.0.0.1:5555" // сервер отправляет сообщения клиентам

#define CLIENT\_PORT "tcp://127.0.0.1:5556" // клиенты отправляет сообщения серверу

#define GAME\_PORT "tcp://127.0.0.1:5557" // клиенты отправляет сообщения серверу

#define MAX\_COUNT\_OF\_GAMES 100

#define MAX\_MAN\_IN\_GAME 100

#define MAX\_NAME\_OF\_PLAYER 30

#define MAX\_NAME\_OF\_GAME 40

char message[1000000];

int main(){

struct Dictionary my\_dict = createDictionary();

char \*stringsClients[100000];

int count\_client = 0;

void \*context = zmq\_ctx\_new(); // Контекст

void \*publisher = zmq\_socket(context, ZMQ\_PUB); // Сокет для отправки сообщений

zmq\_bind(publisher, SERVER\_PORT); // Привязываем сокет к адресу

// Создаем сокет для получения сообщений от клиента

void \*clientSubscriber = zmq\_socket(context, ZMQ\_PULL);

zmq\_bind(clientSubscriber, CLIENT\_PORT); // Привязываем сокет к адресу

// Создаем сокет для получения сообщений от игры

// Я решил сделать через pull и push, потому что не мог делать много публикаторов и один подписчик

// По итогу можно сделать так, только делать zmq\_bind от ZMQ\_SUB именно здесь, а в клиенте делать zmq\_connect от ZMQ\_PUB

void \*gameSubscriber = zmq\_socket(context, ZMQ\_PULL);

zmq\_bind(gameSubscriber, GAME\_PORT); // Привязываем сокет к адресу

// стуктура для сообщений от client и game

// нужно для того, чтобы в цикле не ждать прихода сообщений, а проверять без блокировки, если что-то или нет

// эту проблему можно избежать просто используя один порт, но такое плохо масшатибурется

zmq\_pollitem\_t items\_for\_sockets[] = {

{ clientSubscriber, 0, ZMQ\_POLLIN, 0 },

{ gameSubscriber, 0, ZMQ\_POLLIN, 0 }

};

char buffer\_client[100000];

char buffer\_game[100000];

char command[50]; // хранение команды

char lastMessage[100000];

char nextValue[1000];

char name\_of\_game[50];

char name\_of\_client[50];

while (1) { // работает пока не будет введен eof и не ждет, пока что-то введут!

int rc = zmq\_poll(items\_for\_sockets, 2, 0); // Неблокирующий вызов, возвращает, на скольких сокетах произошли изменения

// if (rc > 0){}

// memset(buffer\_client, 0, sizeof(buffer\_client)); // очищаем buffer\_client

if (items\_for\_sockets[0].revents & ZMQ\_POLLIN){ // проверяем были ли изменения в этом сокете

printf("DEBUG SERVER: get client message\n");

memset(buffer\_client, 0, sizeof(buffer\_client)); // очищаем buffer\_client

zmq\_recv(clientSubscriber, buffer\_client, sizeof(buffer\_client), 0); // Принятие сообщения

// if (strcmp(buffer\_client, lastMessage) == 0){

// printf("DEBUG SERVER: %s \n", buffer\_client);

// continue;

// }

memset(command, 0, sizeof(command));

sscanf(buffer\_client, "%s", command);

if (strcmp(command, "create") == 0){

memset(name\_of\_game, 0, sizeof(name\_of\_game));

memset(name\_of\_client, 0, sizeof(name\_of\_client));

sscanf(buffer\_client, "%\*s %s %s", name\_of\_game, name\_of\_client);

int found = keyExists(&my\_dict, name\_of\_game);

if(found){

memset(message, 0, sizeof(message));

sprintf(message, "Private %s %s CreateNotSuccess", name\_of\_client, name\_of\_game); // создаем строку message

zmq\_send(publisher, message, strlen(message), 0);

}

else{

pid\_t id = fork();

if (id == 0){

execl("./game", "./game", name\_of\_game, NULL);

perror("execl");

}

addToDictionary(&my\_dict, name\_of\_game, name\_of\_client);

memset(message, 0, sizeof(message));

sprintf(message, "Private %s %s CreateSuccess", name\_of\_client, name\_of\_game); // создаем строку message

zmq\_send(publisher, message, strlen(message), 0);

}

}

else if (strcmp(command, "connect") == 0){

memset(name\_of\_game, 0, sizeof(name\_of\_game));

memset(name\_of\_client, 0, sizeof(name\_of\_client));

sscanf(buffer\_client, "%\*s %s %s", name\_of\_game, name\_of\_client);

int found = keyExists(&my\_dict, name\_of\_game);

printf("DEBUG SERVER: try to connect %s, game: %s; found: %d;\n", name\_of\_client, name\_of\_game, found);

if(found){

memset(message, 0, sizeof(message));

sprintf(message, "Private %s %s ConnectSuccess", name\_of\_client, name\_of\_game); // создаем строку message

zmq\_send(publisher, message, strlen(message), 0);

addToDictionary(&my\_dict, name\_of\_game, name\_of\_client);

}

else{

memset(message, 0, sizeof(message));

sprintf(message, "Private %s %s ConnectNotSuccess", name\_of\_client, name\_of\_game); // создаем строку message

zmq\_send(publisher, message, strlen(message), 0);

}

}

else if (strcmp(command, "TryAnswer") == 0){ //проверяем предположеие игрока

zmq\_send(publisher, buffer\_client, strlen(buffer\_client), 0); // отправляем его игре

printf("DEBUG SERVER: try answer massege: %s;\n", buffer\_client);

}

else if (strcmp(command, "InitName") == 0){ // проверяем, существует ли такое имя у игрока

memset(name\_of\_client, 0, sizeof(name\_of\_client));

sscanf(buffer\_client, "%\*s %s", name\_of\_client);

int found = 0; // Флаг для указания наличия строки

for (int i = 0; i < count\_client + 1; ++i) {

if (stringsClients[i] != NULL && strcmp(stringsClients[i], name\_of\_client) == 0) {

found = 1;

break; // Нашли совпадение, выходим из цикла

}

}

if (found){

memset(message, 0, sizeof(message));

sprintf(message, "AnswerName %s repeat", name\_of\_client); // создаем строку message

zmq\_send(publisher, message, strlen(message), 0);

}

else{

memset(message, 0, sizeof(message));

sprintf(message, "AnswerName %s okey", name\_of\_client); // создаем строку message

zmq\_send(publisher, message, strlen(message), 0);

stringsClients[count\_client] = strdup(name\_of\_client);

count\_client ++;

}

}

else if(strcmp(command, "KillServer") == 0){

int keyyy;

sscanf(buffer\_client, "%\*s %d", &keyyy);

if (keyyy == 123456){

memset(message, 0, sizeof(message));

sprintf(message, "ServerWasKilled"); // создаем строку message

zmq\_send(publisher, message, strlen(message), 0);

break;

}

}

else if(strcmp(command, "LeaveGame") == 0){

memset(name\_of\_game, 0, sizeof(name\_of\_game));

memset(name\_of\_client, 0, sizeof(name\_of\_client));

sscanf(buffer\_client, "%\*s %s %s", name\_of\_game, name\_of\_client);

//дальше отправляем клиенту его ход

memset(nextValue, 0, sizeof(nextValue));

// nextValue = getNextValue(&my\_dict, name\_of\_game, name\_of\_client);

strcpy(nextValue, getNextValue(&my\_dict, name\_of\_game, name\_of\_client));

if (nextValue != NULL){

memset(message, 0, sizeof(message));

sprintf(message, "LeaveGAME %s %s", name\_of\_game, name\_of\_client); // создаем строку message

zmq\_send(publisher, message, strlen(message), 0);

printf("DEBUG SERVER: nextValue: %s;\n", nextValue);

memset(message, 0, sizeof(message));

sprintf(message, "YourTurn %s %s", name\_of\_game, nextValue); // создаем строку message

zmq\_send(publisher, message, strlen(message), 0);

}

removePersonFromGameDictionary(&my\_dict, name\_of\_game, name\_of\_client);

}

strcpy(lastMessage, buffer\_client);

printf("DEBUG SERVER: lastMessage: %s; buffer\_client: %s;\n", lastMessage, buffer\_client);

}

if (items\_for\_sockets[1].revents & ZMQ\_POLLIN){

printf("DEBUG SERVER: get game message\n");

memset(buffer\_game, 0, sizeof(buffer\_game)); // очищаем buffer\_game

zmq\_recv(gameSubscriber, buffer\_game, sizeof(buffer\_game), 0);

if (strcmp(buffer\_game, lastMessage) == 0)

continue;

memset(command, 0, sizeof(command));

sscanf(buffer\_game, "%s", command);

if (strcmp(command, "Checked") == 0){

zmq\_send(publisher, buffer\_game, strlen(buffer\_game), 0);

sscanf(buffer\_game, "%\*s %s %s", name\_of\_game, name\_of\_client);

//дальше отправляем клиенту его ход

memset(nextValue, 0, sizeof(nextValue));

// nextValue = getNextValue(&my\_dict, name\_of\_game, name\_of\_client);

strcpy(nextValue, getNextValue(&my\_dict, name\_of\_game, name\_of\_client));

if (nextValue != NULL){

memset(message, 0, sizeof(message));

sprintf(message, "YourTurn %s %s", name\_of\_game, nextValue); // создаем строку message

zmq\_send(publisher, message, strlen(message), 0);

}

}

else if(strcmp(command, "Win") == 0){

zmq\_send(publisher, buffer\_game, strlen(buffer\_game), 0);

sscanf(buffer\_game, "%\*s %s", name\_of\_game);

removeFromDictionary(&my\_dict, name\_of\_game);

}

strcpy(lastMessage, buffer\_game);

// printf("DEBUG SERVER: lastMessage: %s; buffer\_game: %s;\n", lastMessage, buffer\_game);

}

}

zmq\_close(publisher);

zmq\_close(clientSubscriber);

zmq\_close(gameSubscriber);

zmq\_ctx\_destroy(context);

// free(stringsClients);

}

**client.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <fcntl.h>

#include <sys/wait.h>

#include <zmq.h> // gcc client.c -o client -lzmq

#define SERVER\_PORT "tcp://127.0.0.1:5555" // сервер отправляет сообщения клиентам

#define CLIENT\_PORT "tcp://127.0.0.1:5556" // клиенты отправляет сообщения серверу

char my\_name[100];

int can\_write = 1;

int in\_a\_game = 0;

char name\_my\_game[100];

int main(){

int id\_message = 0;

void \*context = zmq\_ctx\_new(); // Контекст

void \*serverSubscriber = zmq\_socket(context, ZMQ\_SUB); // Сокет для принятия сообщений

zmq\_connect(serverSubscriber, SERVER\_PORT); // Подключаемся к адресу

zmq\_setsockopt(serverSubscriber, ZMQ\_SUBSCRIBE, "", 0); // Подписываемся на все сообщения (пустая строка)

void \*publisher = zmq\_socket(context, ZMQ\_PUSH); // Сокет для отправки сообщений

zmq\_connect(publisher, CLIENT\_PORT); // Привязываем сокет к адресу

char buffer[1024]; // Буфер для принятого сообщения

char message[10000];

char command[100];// хранение команды

char command\_serv[100];

char possible\_name[100];

char possible\_name\_game[100];

char result[100];

char answer[100];

char input[100000];

printf("Write your name, please\n");

if (fgets(input, sizeof(input), stdin) == NULL) { // Считываем вводную строку (NULL)

printf("adios");

exit(0);

}

sscanf(input, "%s", my\_name);

int check\_name = 0;

while (check\_name == 0){

memset(buffer, 0, sizeof(buffer)); // очищаем buffer

memset(message, 0, sizeof(message));

memset(command, 0, sizeof(command));

memset(possible\_name, 0, sizeof(possible\_name));

memset(result, 0, sizeof(result));

memset(input, 0, sizeof(input));

sprintf(message, "InitName %s %d", my\_name, id\_message);

zmq\_send(publisher, message, strlen(message), 0); // отправили имя

id\_message++;

// printf("DEBUG: waiting answer\n");

zmq\_recv(serverSubscriber, buffer, sizeof(buffer), 0); // ждем подтверждение

// printf("DEBUG: we get answer\n");

sscanf(buffer, "%s %s %s", command, possible\_name, result); // Считываем начальное слово в command

if ((strcmp(command, "AnswerName") == 0) && (strcmp(possible\_name, my\_name) == 0)){

// printf("%s %s %s \n", command, possible\_name, result);

// printf("%s\n",possible\_name);

// printf("%s\n", my\_name);

if (strcmp(result, "okey") == 0){

printf("Okey, lets play\n");

check\_name = 1;

}

else if (strcmp(result, "repeat") == 0){

printf("Sorry, this name is already exist, try again:\n");

memset(my\_name, 0, sizeof(my\_name));

memset(input, 0, sizeof(input));

if (fgets(input, sizeof(input), stdin) == NULL) { // Считываем вводную строку (NULL)

printf("adios\n");

exit(0);

}

sscanf(input, "%s", my\_name);

}

}

else if(strcmp(command, "ServerWasKilled") == 0){

printf("Sorry, server doesnt work, goodbye\n");

break;

}

else{

// printf("something wrong:\n");

// printf("DEBUG: command: '%s'; possible\_name: '%s'; my\_name: '%s'\n", command, possible\_name, my\_name);

}

}

printf("You can write this:\n newgame [name of game] - create new game\n connect [name of game] - connect to another game\n leave - if you want to leave the game\n");

while (1) {

memset(buffer, 0, sizeof(buffer)); // очищаем buffer

memset(message, 0, sizeof(message));

memset(command, 0, sizeof(command));

memset(command\_serv, 0, sizeof(command\_serv));

memset(possible\_name, 0, sizeof(possible\_name));

memset(possible\_name\_game, 0, sizeof(possible\_name\_game));

memset(result, 0, sizeof(result));

memset(input, 0, sizeof(input));

memset(answer, 0, sizeof(answer));

if(in\_a\_game == 0){ // еcли не в игре

if (can\_write){

if (fgets(input, sizeof(input), stdin) == NULL) { // Считываем вводную строку (NULL)

printf("adios\n");

exit(0);

}

sscanf(input, "%s", command); //читаем команду

if(strcmp(command, "newgame") == 0){

sscanf(input, "%\*s %s", result);

memset(message, 0, sizeof(message));

sprintf(message, "create %s %s %d", result, my\_name, id\_message); // создаем строку message

zmq\_send(publisher, message, strlen(message), 0);

id\_message++;

can\_write = 0;

}

else if(strcmp(command, "connect") == 0){

sscanf(input, "%\*s %s", result);

memset(message, 0, sizeof(message));

sprintf(message, "connect %s %s %d", result, my\_name, id\_message); // создаем строку message

// printf("DEBUG CLIENT: massage for connect: %s;\n", message);

zmq\_send(publisher, message, strlen(message), 0);

id\_message++;

can\_write = 0;

}

else if(strcmp(command, "killserver") == 0){

int keyyy;

sscanf(input, "%\*s %d", &keyyy);

memset(message, 0, sizeof(message));

sprintf(message, "KillServer %d", keyyy); // создаем строку message

// printf("DEBUG CLIENT: massage for connect: %s;\n", message);

zmq\_send(publisher, message, strlen(message), 0);

id\_message++;

}

}

else{

zmq\_recv(serverSubscriber, buffer, sizeof(buffer), 0);

sscanf(buffer, "%s %s %s %s", command\_serv, possible\_name, possible\_name\_game, answer); //читаем команду

// printf("DEBUG CLIENT: message from server NOT in game: %s;\n", buffer);

if (strcmp(command\_serv, "Private") == 0 && strcmp(possible\_name, my\_name) == 0){

if (strcmp(answer, "CreateSuccess") == 0){

printf("You are in the game!\n");

in\_a\_game = 1;

can\_write = 1;

strcpy(name\_my\_game, possible\_name\_game);

// printf("DEBUG CLIENT: name of game: %s; \n", name\_my\_game);

}

else if (strcmp(answer, "CreateNotSuccess") == 0){

printf("This game already exist\n");

can\_write = 1;

}

else if (strcmp(answer, "ConnectSuccess") == 0){

printf("You are in the game!\n");

in\_a\_game = 1;

can\_write = 0;

printf("Wait your turn\n");

strcpy(name\_my\_game, possible\_name\_game);

}

else if (strcmp(answer, "ConnectNotSuccess") == 0){

printf("This game doesnt exist\n");

can\_write = 1;

}

}

else if(strcmp(command, "ServerWasKilled") == 0){

printf("Sorry, server doesnt work, goodbye\n");

break;

}

}

}

else{

if(can\_write){

printf("Please write your answer:\n");

if (fgets(input, sizeof(input), stdin) == NULL) { // Считываем вводную строку (NULL)

printf("adios\n");

exit(0);

}

sscanf(input, "%s", result);

// если хочет покинуть игру

if (strcmp(result, "leave") == 0){

memset(message, 0, sizeof(message));

sprintf(message, "LeaveGame %s %s %d", name\_my\_game, my\_name, id\_message); // создаем строку message

// printf("DEBUG CLIENT: message try answer: %s;\n", message);

zmq\_send(publisher, message, strlen(message), 0);

id\_message++;

in\_a\_game = 0;

can\_write = 1;

printf("You leave the game\nPlease, create new game or connect\n");

continue;

}

memset(message, 0, sizeof(message));

sprintf(message, "TryAnswer %s %s %s %d", name\_my\_game, my\_name, result, id\_message); // создаем строку message

// printf("DEBUG CLIENT: message try answer: %s;\n", message);

zmq\_send(publisher, message, strlen(message), 0);

id\_message++;

can\_write = 0;

}

else{

memset(buffer, 0, sizeof(buffer));

zmq\_recv(serverSubscriber, buffer, sizeof(buffer), 0);

// printf("DEBUG CLIENT: message from server in game: %s;\n", buffer);

memset(command, 0, sizeof(command));

sscanf(buffer, "%s", command);

if (strcmp(command, "Checked") == 0){

// printf("DEBUG CLIENT: check answer good\n");

int cows;

int bulls;

sscanf(buffer, "%\*s %s %s %s %d %d", possible\_name\_game, possible\_name, result, &cows, &bulls);

// printf("User %s try: %s. Answer: cows: %d, bulls: %d\n", possible\_name, result, cows, bulls);

// printf("%s||%s\n", name\_my\_game, possible\_name\_game);

if (strcmp(possible\_name\_game, name\_my\_game) == 0)

printf("User %s try: %s. Answer: cows: %d, bulls: %d\n", possible\_name, result, cows, bulls);

}

else if (strcmp(command, "Win") == 0){

int cows;

int bulls;

sscanf(buffer, "%\*s %s %s %s %d %d",possible\_name\_game, possible\_name, result, &cows, &bulls);

if (strcmp(possible\_name\_game, name\_my\_game) == 0){

if (strcmp(possible\_name, my\_name) == 0)

printf("You are win!\n");

else

printf("User %s win with try: %s\n", possible\_name, result);

in\_a\_game = 0;

can\_write = 1;

printf("Please, create new game or connect\n");

}

}

else if(strcmp(command, "YourTurn") == 0){

sscanf(buffer, "%\*s %s %s",possible\_name\_game, possible\_name);

if (strcmp(possible\_name\_game, name\_my\_game) == 0 && strcmp(possible\_name, my\_name) == 0){

can\_write = 1;

}

else if( strcmp(possible\_name\_game, name\_my\_game) == 0 ){

printf("Waiting player: %s\n", possible\_name);

}

}

else if(strcmp(command, "LeaveGAME") == 0){

sscanf(buffer, "%\*s %s %s",possible\_name\_game, possible\_name);

if(strcmp(possible\_name\_game, name\_my\_game) == 0 ){

printf("Player '%s' leave the game\n", possible\_name);

}

}

else if(strcmp(command, "ServerWasKilled") == 0){

printf("Sorry, server doesnt work, goodbye\n");

break;

}

}

}

}

zmq\_close(publisher);

zmq\_close(serverSubscriber);

zmq\_ctx\_destroy(context);

}

**game.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include <fcntl.h>

#include <sys/wait.h>

#include <zmq.h> // gcc game.c -o game -lzmq

char word[] = "abcd";

char my\_name[100];

#define GAME\_PORT "tcp://127.0.0.1:5557" // клиенты отправляет сообщения серверу

#define SERVER\_PORT "tcp://127.0.0.1:5555" // сервер отправляет сообщения клиентам

void generateRandomString() {

// Символы, которые могут быть использованы в строке

const char charset[] = "abcdefghijklmnopqrstuvwxyz";

// Инициализация генератора случайных чисел

srand((unsigned int)time(NULL));

// Генерация случайных символов

for (int i = 0; i < 4; ++i) {

int index = rand() % (sizeof(charset) - 1);

word[i] = charset[index];

}

// Добавляем завершающий символ '\0'

// result[4] = '\0';

}

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

int id\_message = 0;

generateRandomString();

printf("DEBUG GAME: answer is %s\n", word);

if (argc > 1) {

// Копируем переданное имя в глобальную строку

strcpy(my\_name, argv[1]);

}

void \*context = zmq\_ctx\_new(); // Контекст

void \*serverSubscriber = zmq\_socket(context, ZMQ\_SUB); // Сокет для принятия сообщений

zmq\_connect(serverSubscriber, SERVER\_PORT); // Подключаемся к адресу

zmq\_setsockopt(serverSubscriber, ZMQ\_SUBSCRIBE, "", 0); // Подписываемся на все сообщения (пустая строка)

void \*publisher = zmq\_socket(context, ZMQ\_PUSH); // Сокет для отправки сообщений

zmq\_connect(publisher, GAME\_PORT); // Привязываем сокет к адресу

char buffer[100000];

char command[50];

char result[10];

char name\_man[101];

char possible\_name[100];

// char lastMessage[100000];

char message[10000];

while(1){

memset(buffer, 0, sizeof(buffer)); // очищаем buffer

memset(command, 0, sizeof(command)); // очищаем command

memset(result, 0, sizeof(result));

memset(name\_man, 0, sizeof(name\_man));

memset(possible\_name, 0, sizeof(possible\_name));

memset(message, 0, sizeof(message));

// printf("DEBUG GAME: wait massege \n");

zmq\_recv(serverSubscriber, buffer, sizeof(buffer), 0);

// printf("DEBUG GAME: get massege \n");

// if (strcmp(buffer, lastMessage) == 0){

// continue;

// }

sscanf(buffer, "%s", command); // Считываем начальное слово в command

printf("DEBUG GAME: buffer:%s;\n", buffer);

if (strcmp(command, "TryAnswer") == 0){ // проверяем к нам ли обращаются

sscanf(buffer, "%\*s %s", possible\_name); // Считываем начальное слово в command

// printf("DEBUG GAME: command: %s; possible name: %s; my\_name: %s;\n", command, possible\_name, my\_name);

// printf("DEBUG GAME: checked name \n");

if (strcmp(possible\_name, my\_name) == 0){

// printf("DEBUG GAME: lastMessage: %s;\n", lastMessage);

// strcpy(lastMessage, buffer);

// printf("DEBUG GAME: lastMessage: %s;\n", lastMessage);

printf("DEBUG GAME: try answer \n");

sscanf(buffer, "%\*s %\*s %s %s", name\_man, result);

if (strcmp(result, word) == 0) {

printf("DEBUG GAME: send to winner \n");

sprintf(message, "Win %s %s %s %d", my\_name, name\_man, result, id\_message);

zmq\_send(publisher, message, strlen(message), 0);

id\_message++;

break;

}

int cows = 0;

int bulls = 0;

// Подсчет быков

for (int i = 0; i < strlen(word); ++i) {

if (word[i] == result[i]) {

bulls ++;

}

}

// Подсчет коров

for (int i = 0; i < strlen(word); ++i) {

for (int j = 0; j < strlen(result); ++j) {

if (i != j && word[i] == result[j]) {

cows ++;

break;

}

}

}

sprintf(message, "Checked %s %s %s %d %d %d", my\_name, name\_man, result, cows, bulls, id\_message);

zmq\_send(publisher, message, strlen(message), 0);

id\_message++;

}

}

else if(strcmp(command, "ServerWasKilled") == 0){

break;

}

}

zmq\_close(publisher);

zmq\_close(serverSubscriber);

zmq\_ctx\_destroy(context);

}

**my\_dict.h**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Максимальное количество значений в массиве

#define MAX\_VALUES 100

// Структура для представления записи в словаре

struct KeyValuePair {

char key[50]; // Ключ (строка)

char values[MAX\_VALUES][50]; // Массив значений (строк)

int valueCount; // Текущее количество значений в массиве

};

// Структура словаря

struct Dictionary {

struct KeyValuePair entries[100]; // Массив записей в словаре

int entryCount; // Текущее количество записей в словаре

};

struct Dictionary createDictionary() {

struct Dictionary dictionary;

dictionary.entryCount = 0;

return dictionary;

}

// Функция для добавления значения по ключу

void addToDictionary(struct Dictionary \*dictionary, const char \*key, const char \*value) {

// Ищем запись с таким ключом

for (int i = 0; i < dictionary->entryCount; ++i) {

if (strcmp(dictionary->entries[i].key, key) == 0) {

// Нашли запись с таким ключом

// Добавляем значение в массив

if (dictionary->entries[i].valueCount < MAX\_VALUES) {

strcpy(dictionary->entries[i].values[dictionary->entries[i].valueCount], value);

dictionary->entries[i].valueCount++;

} else {

printf("Превышено максимальное количество значений для ключа %s\n", key);

}

return;

}

}

// Если запись с таким ключом не найдена, создаем новую запись

if (dictionary->entryCount < 100) {

strcpy(dictionary->entries[dictionary->entryCount].key, key);

strcpy(dictionary->entries[dictionary->entryCount].values[0], value);

dictionary->entries[dictionary->entryCount].valueCount = 1;

dictionary->entryCount++;

} else {

printf("Превышено максимальное количество записей в словаре\n");

}

}

// Функция для поиска ключа и возвращения значения по ключу

const char \*findInDictionary(const struct Dictionary \*dictionary, const char \*key) {

for (int i = 0; i < dictionary->entryCount; ++i) {

if (strcmp(dictionary->entries[i].key, key) == 0) {

// Нашли запись с таким ключом

// Возвращаем первое значение из массива (если оно есть)

if (dictionary->entries[i].valueCount > 0) {

return dictionary->entries[i].values[0];

} else {

return "Нет значений для данного ключа";

}

}

}

// Если запись с таким ключом не найдена

return "Ключ не найден";

}

// Функция для проверки наличия ключа в словаре

int keyExists(const struct Dictionary \*dictionary, const char \*key) {

for (int i = 0; i < dictionary->entryCount; ++i) {

if (strcmp(dictionary->entries[i].key, key) == 0) {

// Нашли запись с таким ключом

return 1; // Возвращаем 1, если ключ найден

}

}

// Если запись с таким ключом не найдена

return 0; // Возвращаем 0, если ключ не найден

}

// Функция для добавления ключа в словарь

void addKeyToDictionary(struct Dictionary \*dictionary, const char \*key) {

// Проверяем, существует ли уже запись с таким ключом

if (dictionary->entryCount < 100) {

// Если запись с таким ключом не найдена, создаем новую запись

strcpy(dictionary->entries[dictionary->entryCount].key, key);

dictionary->entryCount++;

} else {

printf("Превышено максимальное количество записей в словаре\n");

}

}

// Функция для поиска следующей строки в массиве по ключу

char\* getNextValue(struct Dictionary\* dictionary, const char\* key, const char\* currentValue) {

for (int i = 0; i < dictionary->entryCount; i++) {

if (strcmp(dictionary->entries[i].key, key) == 0) {

for (int j = 0; j < dictionary->entries[i].valueCount; j++) {

// Ищем текущее значение в массиве

if (strcmp(dictionary->entries[i].values[j], currentValue) == 0) {

// Возвращаем следующее значение (или первое, если текущее последнее)

return dictionary->entries[i].values[(j + 1) % dictionary->entries[i].valueCount];

}

}

}

}

return NULL; // Если ключ или значение не найдены

}

void removeFromDictionary(struct Dictionary \*dictionary, const char \*key) {

for (int i = 0; i < dictionary->entryCount; ++i) {

if (strcmp(dictionary->entries[i].key, key) == 0) {

// Нашли запись с ключом, удаляем её

for (int j = i; j < dictionary->entryCount - 1; ++j) {

// Сдвигаем оставшиеся записи влево

strcpy(dictionary->entries[j].key, dictionary->entries[j + 1].key);

memcpy(dictionary->entries[j].values, dictionary->entries[j + 1].values, sizeof(dictionary->entries[j].values));

dictionary->entries[j].valueCount = dictionary->entries[j + 1].valueCount;

}

// Уменьшаем количество записей

dictionary->entryCount--;

break;

}

}

}

void removePersonFromGameDictionary(struct Dictionary \*dictionary, const char \*key, const char \*valueToRemove) {

for (int i = 0; i < dictionary->entryCount; ++i) {

if (strcmp(dictionary->entries[i].key, key) == 0) {

// Нашли запись с указанным ключом

for (int j = 0; j < dictionary->entries[i].valueCount; ++j) {

if (strcmp(dictionary->entries[i].values[j], valueToRemove) == 0) {

// Нашли строку для удаления из массива значений

// Сдвигаем оставшиеся элементы массива

for (int k = j; k < dictionary->entries[i].valueCount - 1; ++k) {

strcpy(dictionary->entries[i].values[k], dictionary->entries[i].values[k + 1]);

}

// Уменьшаем счетчик значений

dictionary->entries[i].valueCount--;

printf("String '%s' removed from key '%s'\n", valueToRemove, key);

return;

}

}

}

}

printf("String '%s' not found for key '%s'\n", valueToRemove, key);

}

**Протокол работы программы**

**Тестирование:**

$ ./client

Write your name, please

krak

Okey, lets play

You can write this:

newgame [name of game] - create new game

connect [name of game] - connect to another game

leave - if you want to leave the game

connect one

You are in the game!

Wait your turn

User nesty try: qwea. Answer: cows: 1, bulls: 0

Please write your answer:

eada

User krak try: eada. Answer: cows: 0, bulls: 0

Waiting player: nesty

Player 'nesty' leave the game

Please write your answer:

qwea

User krak try: qwea. Answer: cows: 1, bulls: 0

Please write your answer:

qeqa

User krak try: qeqa. Answer: cows: 0, bulls: 0

Please write your answer:

qw

User krak try: qw. Answer: cows: 1, bulls: 0

Please write your answer:

zhbn

User krak try: zhbn. Answer: cows: 0, bulls: 1

Please write your answer:

ghfw

You are win!

Please, create new game or connect

connect one

You are in the game!

Wait your turn

User nesty try: abcd. Answer: cows: 1, bulls: 0

Waiting player: dmit

User dmit try: qwer. Answer: cows: 0, bulls: 0

Please write your answer:

eeer

User krak try: eeer. Answer: cows: 0, bulls: 0

Waiting player: anast

User anast try: trsd. Answer: cows: 2, bulls: 0

Waiting player: nesty

Player 'nesty' leave the game

Waiting player: dmit

User dmit win with try: gtdo

Please, create new game or connect

adios

**Часть Strace, полный в strace\_kp.txt:**

strace -f ./client

execve("./client", ["./client"], 0x7ffec44445e8 /\* 25 vars \*/) = 0

brk(NULL) = 0x55a07439b000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fff8a86b6f0) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f9a0c3f7000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=19839, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 19839, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f9a0c3f2000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\240\233\1\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=634936, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 636784, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f9a0c356000

mmap(0x7f9a0c36e000, 397312, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x18000) = 0x7f9a0c36e000

mmap(0x7f9a0c3cf000, 106496, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x79000) = 0x7f9a0c3cf000

mmap(0x7f9a0c3e9000, 36864, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x92000) = 0x7f9a0c3e9000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0 =\340\2563\265?\356\25x\261\27\313A#\350"..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...}, AT\_EMPTY\_PATH) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

mmap(NULL, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f9a0c12e000

mmap(0x7f9a0c156000, 1658880, PROT\_READ|PROT\_EXEC, D) = 0

…

mprotect(0x7f9a0baa3000, 4096, PROT\_READ) = 0

mprotect(0x7f9a0baa9000, 4096, PROT\_READ) = 0

mprotect(0x7f9a0bad7000, 4096, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7f9a0c3f2000, 19839) = 0

getrandom("\x04\x58\xf0\x9c\x70\xce\xbd\x9e", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55a07439b000

brk(0x55a0743bc000) = 0x55a0743bc000

openat(AT\_FDCWD, "/sys/devices/system/cpu/online", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "0-7\n", 1024) = 4

close(3) = 0

openat(AT\_FDCWD, "/sys/devices/system/cpu", O\_RDONLY|O\_NONBLOCK|O\_CLOEXEC|O\_DIRECTORY) = 3

newfstatat(3, "", {st\_mode=S\_IFDIR|0755, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

getdents64(3, 0x55a0743acee0 /\* 23 entries \*/, 32768) = 656

getdents64(3, 0x55a0743acee0 /\* 0 entries \*/, 32768) = 0

close(3) = 0

getpid() = 171301

sched\_getaffinity(171301, 128, [0, 1, 2, 3, 4, 5, 6, 7]) = 32

newfstatat(AT\_FDCWD, "/etc/nsswitch.conf", {st\_mode=S\_IFREG|0644, st\_size=510, ...}, 0) = 0

…

openat(AT\_FDCWD, "/lib/libnss\_db-2.35.so", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/lib/libnss\_db-2.35.so", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

munmap(0x7f9a0c3f2000, 19839) = 0

openat(AT\_FDCWD, "/etc/protocols", O\_RDONLY|O\_CLOEXEC) = 3

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=2932, ...}, AT\_EMPTY\_PATH) = 0

lseek(3, 0, SEEK\_SET) = 0

read(3, "# Internet (IP) protocols\n#\n# Up"..., 4096) = 2932

read(3, "", 4096) = 0

close(3) = 0

eventfd2(0, EFD\_CLOEXEC) = 3

fcntl(3, F\_GETFL) = 0x2 (flags O\_RDWR)

fcntl(3, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

fcntl(3, F\_GETFL) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

fcntl(3, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

getpid() = 171301

getpid() = 171301

…

[pid 171303] fcntl(10, F\_GETFL <unfinished ...>

[pid 171301] newfstatat(1, "", <unfinished ...>

[pid 171303] <... fcntl resumed>) = 0x2 (flags O\_RDWR)

[pid 171301] <... newfstatat resumed>{st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x7), ...}, AT\_EMPTY\_PATH) = 0

[pid 171303] fcntl(10, F\_SETFL, O\_RDWR|O\_NONBLOCK <unfinished ...>

[pid 171301] write(1, "Write your name, please\n", 24 <unfinished ...>

Write your name, please

[pid 171303] <... fcntl resumed>) = 0

[pid 171301] <... write resumed>) = 24

[pid 171303] connect(10, {sa\_family=AF\_INET, sin\_port=htons(5555), sin\_addr=inet\_addr("127.0.0.1")}, 16 <unfinished ...>

[pid 171301] newfstatat(0, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x7), ...}, AT\_EMPTY\_PATH) = 0

[pid 171301] read(0, <unfinished ...>

[pid 171303] <... connect resumed>) = -1 EINPROGRESS (Operation now in progress)

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_ADD, 10, {events=0, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 10, {events=EPOLLOUT, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] socket(AF\_INET, SOCK\_STREAM|SOCK\_CLOEXEC, IPPROTO\_TCP) = 11

[pid 171303] fcntl(11, F\_GETFL) = 0x2 (flags O\_RDWR)

[pid 171303] fcntl(11, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

[pid 171303] connect(11, {sa\_family=AF\_INET, sin\_port=htons(5556), sin\_addr=inet\_addr("127.0.0.1")}, 16) = -1 EINPROGRESS (Operation now in progress)

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_ADD, 11, {events=0, data={u32=67114064, u64=140299468805200}}) = 0

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLOUT, data={u32=67114064, u64=140299468805200}}) = 0

[pid 171303] getpid() = 171301

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171303] epoll\_wait(7, [{events=EPOLLOUT, data={u32=67114032, u64=140299468805168}}, {events=EPOLLOUT, data={u32=67114064, u64=140299468805200}}], 256, -1) = 2

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_DEL, 10, 0x7f9a04001434) = 0

[pid 171303] getsockopt(10, SOL\_SOCKET, SO\_ERROR, [0], [4]) = 0

[pid 171303] setsockopt(10, SOL\_TCP, TCP\_NODELAY, [1], 4) = 0

[pid 171303] getsockname(10, {sa\_family=AF\_INET, sin\_port=htons(57700), sin\_addr=inet\_addr("127.0.0.1")}, [128 => 16]) = 0

[pid 171303] getpeername(10, {sa\_family=AF\_INET, sin\_port=htons(5555), sin\_addr=inet\_addr("127.0.0.1")}, [128 => 16]) = 0

[pid 171303] fcntl(10, F\_GETFL) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

[pid 171303] fcntl(10, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

[pid 171303] getpid() = 171301

[pid 171303] write(6, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_DEL, 11, 0x7f9a04001454) = 0

[pid 171303] getsockopt(11, SOL\_SOCKET, SO\_ERROR, [0], [4]) = 0

[pid 171303] setsockopt(11, SOL\_TCP, TCP\_NODELAY, [1], 4) = 0

[pid 171303] getsockname(11, {sa\_family=AF\_INET, sin\_port=htons(54700), sin\_addr=inet\_addr("127.0.0.1")}, [128 => 16]) = 0

[pid 171303] getpeername(11, {sa\_family=AF\_INET, sin\_port=htons(5556), sin\_addr=inet\_addr("127.0.0.1")}, [128 => 16]) = 0

[pid 171303] fcntl(11, F\_GETFL) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

[pid 171303] fcntl(11, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

[pid 171303] epoll\_wait(7, [{events=EPOLLIN, data={u32=1950032480, u64=94147633160800}}], 256, -1) = 1

[pid 171303] getpid() = 171301

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0) = 1 ([{fd=6, revents=POLLIN}])

[pid 171303] getpid() = 171301

[pid 171303] read(6, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_ADD, 10, {events=0, data={u32=67114064, u64=140299468805200}}) = 0

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 10, {events=EPOLLIN, data={u32=67114064, u64=140299468805200}}) = 0

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 10, {events=EPOLLIN|EPOLLOUT, data={u32=67114064, u64=140299468805200}}) = 0

[pid 171303] recvfrom(10, "\377\0\0\0\0\0\0\0\1\177", 12, 0, NULL, NULL) = 10

[pid 171303] recvfrom(10, 0x7f9a040022a2, 2, 0, NULL, NULL) = -1 EAGAIN (Resource temporarily unavailable)

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_ADD, 11, {events=0, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN|EPOLLOUT, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] recvfrom(11, "\377\0\0\0\0\0\0\0\1\177", 12, 0, NULL, NULL) = 10

[pid 171303] recvfrom(11, 0x7f9a04002a62, 2, 0, NULL, NULL) = -1 EAGAIN (Resource temporarily unavailable)

[pid 171303] getpid() = 171301

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171303] epoll\_wait(7, [{events=EPOLLOUT, data={u32=67114064, u64=140299468805200}}, {events=EPOLLOUT, data={u32=67114032, u64=140299468805168}}], 256, 29998) = 2

[pid 171303] sendto(10, "\377\0\0\0\0\0\0\0\1\177\3", 11, 0, NULL, 0) = 11

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 10, {events=EPOLLIN, data={u32=67114064, u64=140299468805200}}) = 0

[pid 171303] sendto(11, "\377\0\0\0\0\0\0\0\1\177\3", 11, 0, NULL, 0) = 11

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] epoll\_wait(7, [{events=EPOLLIN, data={u32=67114064, u64=140299468805200}}, {events=EPOLLIN, data={u32=67114032, u64=140299468805168}}], 256, 29998) = 2

[pid 171303] recvfrom(10, "\3\1", 2, 0, NULL, NULL) = 2

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 10, {events=EPOLLIN|EPOLLOUT, data={u32=67114064, u64=140299468805200}}) = 0

[pid 171303] recvfrom(10, "NULL\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 52, 0, NULL, NULL) = 52

[pid 171303] recvfrom(10, 0x7f9a04004bc8, 8192, 0, NULL, NULL) = -1 EAGAIN (Resource temporarily unavailable)

[pid 171303] recvfrom(11, "\3\1", 2, 0, NULL, NULL) = 2

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN|EPOLLOUT, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] recvfrom(11, "NULL\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 52, 0, NULL, NULL) = 52

[pid 171303] recvfrom(11, 0x7f9a0400b8c8, 8192, 0, NULL, NULL) = -1 EAGAIN (Resource temporarily unavailable)

[pid 171303] epoll\_wait(7, [{events=EPOLLOUT, data={u32=67114064, u64=140299468805200}}, {events=EPOLLOUT, data={u32=67114032, u64=140299468805168}}], 256, 29996) = 2

[pid 171303] sendto(10, "\1NULL\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 53, 0, NULL, 0) = 53

[pid 171303] sendto(11, "\1NULL\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 53, 0, NULL, 0) = 53

[pid 171303] epoll\_wait(7, [{events=EPOLLOUT, data={u32=67114064, u64=140299468805200}}, {events=EPOLLOUT, data={u32=67114032, u64=140299468805168}}], 256, 29996) = 2

[pid 171303] sendto(10, "\4\31\5READY\vSocket-Type\0\0\0\3SUB", 27, 0, NULL, 0) = 27

[pid 171303] sendto(11, "\4\32\5READY\vSocket-Type\0\0\0\4PUSH", 28, 0, NULL, 0) = 28

[pid 171303] epoll\_wait(7, [{events=EPOLLIN|EPOLLOUT, data={u32=67114064, u64=140299468805200}}, {events=EPOLLIN|EPOLLOUT, data={u32=67114032, u64=140299468805168}}], 256, 29995) = 2

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 10, {events=EPOLLIN, data={u32=67114064, u64=140299468805200}}) = 0

[pid 171303] recvfrom(10, "\4\31\5READY\vSocket-Type\0\0\0\3PUB", 8192, 0, NULL, NULL) = 27

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 10, {events=EPOLLIN|EPOLLOUT, data={u32=67114064, u64=140299468805200}}) = 0

[pid 171303] sendto(10, "\4\n\tSUBSCRIBE", 12, 0, NULL, 0) = 12

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] recvfrom(11, "\4\32\5READY\vSocket-Type\0\0\0\4PULL", 8192, 0, NULL, NULL) = 28

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN|EPOLLOUT, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] epoll\_wait(7, [{events=EPOLLOUT, data={u32=67114064, u64=140299468805200}}], 256, -1) = 1

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 10, {events=EPOLLIN, data={u32=67114064, u64=140299468805200}}) = 0

[pid 171303] epoll\_wait(7, dmit

<unfinished ...>

[pid 171301] <... read resumed>"dmit\n", 1024) = 5

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=9, events=POLLIN}], 1, 0) = 1 ([{fd=9, revents=POLLIN}])

[pid 171301] getpid() = 171301

[pid 171301] read(9, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171301] getpid() = 171301

[pid 171301] write(6, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=1950032480, u64=94147633160800}}], 256, -1) = 1

[pid 171301] getpid( <unfinished ...>

[pid 171303] getpid( <unfinished ...>

[pid 171301] <... getpid resumed>) = 171301

[pid 171303] <... getpid resumed>) = 171301

[pid 171301] poll([{fd=8, events=POLLIN}], 1, -1 <unfinished ...>

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0 <unfinished ...>

[pid 171301] <... poll resumed>) = 1 ([{fd=8, revents=POLLIN}])

[pid 171303] <... poll resumed>) = 1 ([{fd=6, revents=POLLIN}])

[pid 171301] getpid( <unfinished ...>

[pid 171303] getpid( <unfinished ...>

[pid 171301] <... getpid resumed>) = 171301

[pid 171303] <... getpid resumed>) = 171301

[pid 171301] read(8, <unfinished ...>

[pid 171303] read(6, <unfinished ...>

[pid 171301] <... read resumed>"\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] <... read resumed>"\1\0\0\0\0\0\0\0", 8) = 8

[pid 171301] getpid( <unfinished ...>

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN|EPOLLOUT, data={u32=67114032, u64=140299468805168}} <unfinished ...>

[pid 171301] <... getpid resumed>) = 171301

[pid 171303] <... epoll\_ctl resumed>) = 0

[pid 171301] poll([{fd=8, events=POLLIN}], 1, 0 <unfinished ...>

[pid 171303] sendto(11, "\0\17InitName dmit 0", 17, 0, NULL, 0 <unfinished ...>

[pid 171301] <... poll resumed>) = 0 (Timeout)

[pid 171301] getpid( <unfinished ...>

[pid 171303] <... sendto resumed>) = 17

[pid 171301] <... getpid resumed>) = 171301

[pid 171303] getpid( <unfinished ...>

[pid 171301] poll([{fd=8, events=POLLIN}], 1, -1 <unfinished ...>

[pid 171303] <... getpid resumed>) = 171301

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171303] epoll\_wait(7, [{events=EPOLLOUT, data={u32=67114032, u64=140299468805168}}], 256, -1) = 1

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] epoll\_wait(7, [{events=EPOLLIN, data={u32=67114064, u64=140299468805200}}], 256, -1) = 1

[pid 171303] recvfrom(10, "\0\24AnswerName dmit okey", 8192, 0, NULL, NULL) = 22

[pid 171303] getpid() = 171301

[pid 171303] write(8, "\1\0\0\0\0\0\0\0", 8 <unfinished ...>

[pid 171301] <... poll resumed>) = 1 ([{fd=8, revents=POLLIN}])

[pid 171303] <... write resumed>) = 8

[pid 171303] epoll\_wait(7, <unfinished ...>

[pid 171301] getpid() = 171301

[pid 171301] read(8, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171301] write(1, "Okey, lets play\n", 16Okey, lets play

) = 16

[pid 171301] write(1, "You can write this:\n newgame [na"..., 112You can write this:

newgame [name of game] - create new game

connect [name of game] - connect to another game

) = 112

[pid 171301] write(1, " leave - if you want to leave th"..., 39 leave - if you want to leave the game

) = 39

[pid 171301] read(0, newgame one

"newgame one\n", 1024) = 12

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171301] getpid() = 171301

[pid 171301] write(6, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=1950032480, u64=94147633160800}}], 256, -1) = 1

[pid 171301] getpid() = 171301

[pid 171303] getpid( <unfinished ...>

[pid 171301] poll([{fd=8, events=POLLIN}], 1, -1 <unfinished ...>

[pid 171303] <... getpid resumed>) = 171301

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0) = 1 ([{fd=6, revents=POLLIN}])

[pid 171303] getpid() = 171301

[pid 171303] read(6, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN|EPOLLOUT, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] sendto(11, "\0\21create one dmit 1", 19, 0, NULL, 0) = 19

[pid 171303] getpid() = 171301

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171303] epoll\_wait(7, [{events=EPOLLOUT, data={u32=67114032, u64=140299468805168}}], 256, -1) = 1

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] epoll\_wait(7, [{events=EPOLLIN, data={u32=67114064, u64=140299468805200}}], 256, -1) = 1

[pid 171303] recvfrom(10, "\0\36Private dmit one CreateSuccess", 8192, 0, NULL, NULL) = 32

[pid 171303] getpid() = 171301

[pid 171303] write(8, "\1\0\0\0\0\0\0\0", 8 <unfinished ...>

[pid 171301] <... poll resumed>) = 1 ([{fd=8, revents=POLLIN}])

[pid 171303] <... write resumed>) = 8

[pid 171301] getpid( <unfinished ...>

[pid 171303] epoll\_wait(7, <unfinished ...>

[pid 171301] <... getpid resumed>) = 171301

[pid 171301] read(8, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171301] write(1, "You are in the game!\n", 21You are in the game!

) = 21

[pid 171301] write(1, "Please write your answer:\n", 26Please write your answer:

) = 26

[pid 171301] read(0, <unfinished ...>

[pid 171303] <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=67114064, u64=140299468805200}}], 256, -1) = 1

[pid 171303] recvfrom(10, "\0\25AnswerName anast okey", 8192, 0, NULL, NULL) = 23

[pid 171303] epoll\_wait(7, [{events=EPOLLIN, data={u32=67114064, u64=140299468805200}}], 256, -1) = 1

[pid 171303] recvfrom(10, "\0 Private anast one ConnectSucce"..., 8192, 0, NULL, NULL) = 34

[pid 171303] epoll\_wait(7,aqwe

<unfinished ...>

[pid 171301] <... read resumed>"aqwe\n", 1024) = 5

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171301] getpid() = 171301

[pid 171301] write(6, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=1950032480, u64=94147633160800}}], 256, -1) = 1

[pid 171301] getpid( <unfinished ...>

[pid 171303] getpid( <unfinished ...>

[pid 171301] <... getpid resumed>) = 171301

[pid 171303] <... getpid resumed>) = 171301

[pid 171301] poll([{fd=8, events=POLLIN}], 1, -1 <unfinished ...>

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0) = 1 ([{fd=6, revents=POLLIN}])

[pid 171303] getpid() = 171301

[pid 171303] read(6, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN|EPOLLOUT, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] sendto(11, "\0\31TryAnswer one dmit aqwe 2", 27, 0, NULL, 0) = 27

[pid 171303] getpid() = 171301

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171303] epoll\_wait(7, [{events=EPOLLOUT, data={u32=67114032, u64=140299468805168}}, {events=EPOLLIN, data={u32=67114064, u64=140299468805200}}], 256, -1) = 2

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] recvfrom(10, "\0\31TryAnswer one dmit aqwe 2\0\33Che"..., 8192, 0, NULL, NULL) = 76

[pid 171303] getpid() = 171301

[pid 171303] write(8, "\1\0\0\0\0\0\0\0", 8 <unfinished ...>

[pid 171301] <... poll resumed>) = 1 ([{fd=8, revents=POLLIN}])

[pid 171303] <... write resumed>) = 8

[pid 171301] getpid( <unfinished ...>

[pid 171303] epoll\_wait(7, <unfinished ...>

[pid 171301] <... getpid resumed>) = 171301

[pid 171301] read(8, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171301] write(1, "User dmit try: aqwe. Answer: cow"..., 47User dmit try: aqwe. Answer: cows: 1, bulls: 0

) = 47

[pid 171301] write(1, "Waiting player: anast\n", 22Waiting player: anast

) = 22

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=8, events=POLLIN}], 1, -1 <unfinished ...>

[pid 171303] <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=67114064, u64=140299468805200}}], 256, -1) = 1

[pid 171303] recvfrom(10, "\0\32TryAnswer one anast abcd 2", 8192, 0, NULL, NULL) = 28

[pid 171303] getpid() = 171301

[pid 171303] write(8, "\1\0\0\0\0\0\0\0", 8 <unfinished ...>

[pid 171301] <... poll resumed>) = 1 ([{fd=8, revents=POLLIN}])

[pid 171303] <... write resumed>) = 8

[pid 171301] getpid( <unfinished ...>

[pid 171303] epoll\_wait(7, <unfinished ...>

[pid 171301] <... getpid resumed>) = 171301

[pid 171303] <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=67114064, u64=140299468805200}}], 256, -1) = 1

[pid 171301] read(8, <unfinished ...>

[pid 171303] recvfrom(10, <unfinished ...>

[pid 171301] <... read resumed>"\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] <... recvfrom resumed>"\0\34Checked one anast abcd 0 0 1\0\21"..., 8192, 0, NULL, NULL) = 49

[pid 171301] getpid( <unfinished ...>

[pid 171303] epoll\_wait(7, <unfinished ...>

[pid 171301] <... getpid resumed>) = 171301

[pid 171301] poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171301] write(1, "User anast try: abcd. Answer: co"..., 48User anast try: abcd. Answer: cows: 0, bulls: 0

) = 48

[pid 171301] write(1, "Please write your answer:\n", 26Please write your answer:

) = 26

[pid 171301] read(0, ntqw

"ntqw\n", 1024) = 5

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171301] getpid() = 171301

[pid 171301] write(6, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=1950032480, u64=94147633160800}}], 256, -1) = 1

[pid 171301] getpid( <unfinished ...>

[pid 171303] getpid( <unfinished ...>

[pid 171301] <... getpid resumed>) = 171301

[pid 171303] <... getpid resumed>) = 171301

[pid 171301] poll([{fd=8, events=POLLIN}], 1, -1 <unfinished ...>

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0) = 1 ([{fd=6, revents=POLLIN}])

[pid 171303] getpid() = 171301

[pid 171303] read(6, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN|EPOLLOUT, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] sendto(11, "\0\31TryAnswer one dmit ntqw 3", 27, 0, NULL, 0) = 27

[pid 171303] getpid() = 171301

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171303] epoll\_wait(7, [{events=EPOLLOUT, data={u32=67114032, u64=140299468805168}}, {events=EPOLLIN, data={u32=67114064, u64=140299468805200}}], 256, -1) = 2

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] recvfrom(10, "\0\31TryAnswer one dmit ntqw 3\0\33Che"..., 8192, 0, NULL, NULL) = 76

[pid 171303] getpid() = 171301

[pid 171303] write(8, "\1\0\0\0\0\0\0\0", 8 <unfinished ...>

[pid 171301] <... poll resumed>) = 1 ([{fd=8, revents=POLLIN}])

[pid 171303] <... write resumed>) = 8

[pid 171301] getpid( <unfinished ...>

[pid 171303] epoll\_wait(7, <unfinished ...>

[pid 171301] <... getpid resumed>) = 171301

[pid 171301] read(8, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171301] write(1, "User dmit try: ntqw. Answer: cow"..., 47User dmit try: ntqw. Answer: cows: 2, bulls: 1

) = 47

[pid 171301] write(1, "Waiting player: anast\n", 22Waiting player: anast

) = 22

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=8, events=POLLIN}], 1, -1 <unfinished ...>

[pid 171303] <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=67114064, u64=140299468805200}}], 256, -1) = 1

[pid 171303] recvfrom(10, "\0\32TryAnswer one anast nqwe 3", 8192, 0, NULL, NULL) = 28

[pid 171303] getpid() = 171301

[pid 171303] write(8, "\1\0\0\0\0\0\0\0", 8 <unfinished ...>

[pid 171301] <... poll resumed>) = 1 ([{fd=8, revents=POLLIN}])

[pid 171303] <... write resumed>) = 8

[pid 171301] getpid( <unfinished ...>

[pid 171303] epoll\_wait(7, <unfinished ...>

[pid 171301] <... getpid resumed>) = 171301

[pid 171303] <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=67114064, u64=140299468805200}}], 256, -1) = 1

[pid 171301] read(8, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] recvfrom(10, <unfinished ...>

[pid 171301] getpid( <unfinished ...>

[pid 171303] <... recvfrom resumed>"\0\34Checked one anast nqwe 1 1 3\0\21"..., 8192, 0, NULL, NULL) = 49

[pid 171301] <... getpid resumed>) = 171301

[pid 171303] epoll\_wait(7, <unfinished ...>

[pid 171301] poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171301] write(1, "User anast try: nqwe. Answer: co"..., 48User anast try: nqwe. Answer: cows: 1, bulls: 1

) = 48

[pid 171301] write(1, "Please write your answer:\n", 26Please write your answer:

) = 26

[pid 171301] read(0,njtq

"njtq\n", 1024) = 5

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171301] getpid() = 171301

[pid 171301] write(6, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=1950032480, u64=94147633160800}}], 256, -1) = 1

[pid 171301] getpid() = 171301

[pid 171303] getpid( <unfinished ...>

[pid 171301] poll([{fd=8, events=POLLIN}], 1, -1 <unfinished ...>

[pid 171303] <... getpid resumed>) = 171301

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0) = 1 ([{fd=6, revents=POLLIN}])

[pid 171303] getpid() = 171301

[pid 171303] read(6, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN|EPOLLOUT, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] sendto(11, "\0\31TryAnswer one dmit njtq 4", 27, 0, NULL, 0) = 27

[pid 171303] getpid() = 171301

[pid 171303] poll([{fd=6, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171303] epoll\_wait(7, [{events=EPOLLOUT, data={u32=67114032, u64=140299468805168}}, {events=EPOLLIN, data={u32=67114064, u64=140299468805200}}], 256, -1) = 2

[pid 171303] epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN, data={u32=67114032, u64=140299468805168}}) = 0

[pid 171303] recvfrom(10, "\0\31TryAnswer one dmit njtq 4\0\23Win"..., 8192, 0, NULL, NULL) = 48

[pid 171303] getpid() = 171301

[pid 171303] write(8, "\1\0\0\0\0\0\0\0", 8 <unfinished ...>

[pid 171301] <... poll resumed>) = 1 ([{fd=8, revents=POLLIN}])

[pid 171303] <... write resumed>) = 8

[pid 171301] getpid( <unfinished ...>

[pid 171303] epoll\_wait(7, <unfinished ...>

[pid 171301] <... getpid resumed>) = 171301

[pid 171301] read(8, "\1\0\0\0\0\0\0\0", 8) = 8

[pid 171301] getpid() = 171301

[pid 171301] poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

[pid 171301] write(1, "You are win!\n", 13You are win!

) = 13

[pid 171301] write(1, "Please, create new game or conne"..., 35Please, create new game or connect

) = 35

[pid 171301] read(0, "", 1024) = 0

[pid 171301] write(1, "adios\n", 6adios

) = 6

[pid 171301] exit\_group(0) = ?

[pid 171303] <... epoll\_wait resumed> <unfinished ...>) = ?

[pid 171302] <... epoll\_wait resumed> <unfinished ...>) = ?

[pid 171303] +++ exited with 0 +++

[pid 171302] +++ exited with 0 +++

+++ exited with 0 +++

**Вывод**

В результате выполнения данной лабораторной работы я еще больше погрузился в тему брокеров сообщений. Мне пришлось сильнее углубиться в библиотеку ZMQ. Было трудно следить за всеми потоками сообщений от процессов друг другу. Это, несомненно, поможет мне в будущем писать масштабные клиент-серверы и многое другое.