Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу

«Операционные системы»

Тема работы

«Морской бой на memory map»

Студент: Путилин Д.Н.
Группа: М8О-207Б-21
Вариант: 6
Преподаватель: Миронов Евгений Сергеевич
Оценка: ___________
Дата: ___________
Подпись: ___________

Москва, 2022

Содержание

Репозиторий

https://github.com/putilin21dn/OC

Постановка задачи

Цель работы

Консоль-серверная игра. 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

Задание

Морской бой. Общение между сервером и клиентом необходимо организовать при помощи memory map. Каждый игрок должен при запуске ввести свой логин. Должна быть предоставлена возможность отправить приглашение на игру другому игроку по логину

Общие сведения о программе

MappedFile.hpp - реализация mapped file. Содержит структуру, в которой
хранится файловый дескриптор и массив чаров.
Player_Game.hpp - отдельный файл классов игрока и игры.
server.cpp - реализация программы сервера.
client.cpp - реализация программы клиента.

Общий метод и алгоритм решения

Сначала запускается сервер, после этого два клиента. Один из клиентов будет создателем игры, другой будет к ней присоединяться. Существует два способо соединения клиентов. Первый, это второй пользователь просто вводит название игры и пароль к ней. Второй, это создатель игры отправляет приглашение по логину, затем другой проверяет командой check на наличие приглошения, если есть, то устанавливается соединение.

Исходный код

server.cpp

```
#include <fcntl.h>
#include <pthread.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <unistd.h>
#include <cassert>
#include <cstring>
#include <iostream>
#include <map>
#include <vector>
#include "MappedFile.hpp"
#include "Player_Game.hpp"
```

```
#include <fstream>




int main() {
    // creator, connector - players
    Player creator;
    Player connector;
    Game game;
    MappedFile mapped_file;
    string client_message = "";
    int er;
    mapped_file.fd = shm_open(_BUFFER_NAME, O_RDWR | O_CREAT, _SHM_OPEN_MODE);
    if (mapped_file.fd == -1) {
        perror("sem_open error");
        return -1;
    }
    if (ftruncate(mapped_file.fd, _MAPPED_SIZE) == -1) {
        perror("ftruncate error");
        return -1;
    }

    mapped_file.data = (char *)mmap(NULL, _MAPPED_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED,
mapped_file.fd, 0);
    if (mapped_file.data == MAP_FAILED) {
        perror("mmap error");
        return -1;
    }

    memset(mapped_file.data, '\0', _MAPPED_SIZE);
    pthread_mutex_t mutex;

    if (er = pthread_mutex_init(&mutex, NULL))
    {
        printf("Mutex init error: %d", er);
        return -1;
    }
    cout << "Server is working now! Please start a game and it will be displayed here!" << endl;
    while (true) {
        if (mapped_file.data[0] == EOF) {
            break;
        }
        if (mapped_file.data[0] == '\0') {
            continue;
        }
        if (!(mapped_file.data[0] == 'O' && mapped_file.data[1] == 'N' &&
              mapped_file.data[2] == _MSG_SEP)) {
            continue;
        }
        cout << "Locking mutex" << endl;
        if (pthread_mutex_lock(&mutex) != 0) {
            perror("Error locking mutex\n");
            return -1;
        }
        client_message = mapped_file.data;
        cout << "Has received next message from client: " << client_message << '\n';
        memset(mapped_file.data, '\0', _MAPPED_SIZE);
        vector<string> client_commands;
        string strings = "";
        //write client_command
        for (int i = 0; i < client_message.size(); ++i) {
            if (client_message[i] == _MSG_SEP) {
                client_commands.push_back(strings);
                strings = "";
            }
            else {
                strings.push_back(client_message[i]);
            }
        }

        if (client_commands[2] == "create") {
            if (game.created || game.name == client_commands[3]) {

                string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP + "zero-
places" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                cout << "Sending to client next message: " << player_message << '\n';
            }
            else {
                game.created = true;
                creator.turn = true;
                connector.turn = false;
                creator.username = client_commands[1];
                Map(creator.field);
                // cout << "creator\n";
                // PrintField(creator.field);
                game.name = client_commands[3];
                game.password = client_commands[4];
                string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP + "gamecre-
ated" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                cout << "Sending to client next message: " << player_message << '\n';
            }
        }
        else if (client_commands[2] == "connect") {
            if (game.connected) {
                string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP + "zero-
places" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                cout << "Sending to client next message: " << player_message << '\n';
            }
            else {
                if (game.name == client_commands[3]) {
                    if (game.password == client_commands[4]) {
                        game.connected = true;
                        connector.turn = false;
                        creator.turn = true;
                        connector.username = client_commands[1];
                        Map(connector.field);
                        // cout << "connector\n";
                        // PrintField(connector.field);
                        string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"connected" + _MSG_SEP;
                        sprintf(mapped_file.data, "%s", player_message.c_str());
                        cout << "Sending to client next message: " << player_message << '\n';
                    }
                    else {
                        game.connected = false;
                        string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"wrongpassword" + _MSG_SEP;
                        sprintf(mapped_file.data, "%s", player_message.c_str());
                        cout << "Sending to client next message: " << player_message << '\n';
                    }
                }
                else {
                    game.connected = false;
                    string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"gamenotexists" + _MSG_SEP;
                    sprintf(mapped_file.data, "%s", player_message.c_str());
                    cout << "Sending to client next message:" << player_message << '\n';
                }
            }
        }
        else if (client_commands[2] == "invite"){
            if (game.connected) {
                string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP + "zero-
places" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                cout << "Sending to client next message: " << player_message << '\n';
            }
            else{
                game.name = client_commands[3];
                game.password = client_commands[4];
                connector.invite = true;
                string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP + "invited"
+ _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                cout << "Sending to client next message: " << player_message << '\n';
            }
        }
        else if(client_commands[2] == "check"){
            if(connector.invite){
                game.connected = true;
                connector.turn = false;
                creator.turn = true;
                connector.username = client_commands[1];
                Map(connector.field);
                // cout << "connector\n";
                // PrintField(connector.field);
                string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP + "checked"
+ _MSG_SEP + game.name + _MSG_SEP + game.password + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                cout << "Sending to client next message: " << player_message << '\n';
            }
            else{
                string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"notchecked" + _MSG_SEP + game.name + _MSG_SEP + game.password + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                cout << "Sending to client next message: " << player_message << '\n';
            }
        }
        else if (client_commands[2] == "shoot") {
            if (!game.connected) {
                string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP + "no-
tatgame" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                cout << "Sending to client next message: " << player_message << '\n';
            }
            // shoot connector
            if (client_commands[1] == connector.username) {
                if (connector.turn && !creator.turn) {  // check try
                    if (game.name == client_commands[3]) {
                        int number = stoi(client_commands[5]);
                        string l = client_commands[4];
                        char letter = l[0];
                        // check position maybe
                        if (creator.field[number][int(letter) - int('A') + 1] == 'X' &&
                        (creator.field[number][int(letter) - int('A') + 2] == '.' || creator.-
field[number][int(letter) - int('A') + 2] == 'm' || creator.field[number][int(letter) - int('A') +
2] == 'w') &&
                        (creator.field[number - 1][int(letter) - int('A') + 1] == '.' || creator.-
field[number - 1][int(letter) - int('A') + 1] == 'm' || creator.field[number - 1][int(letter) -
int('A') + 1] == 'w') &&
                        (creator.field[number - 1][int(letter) - int('A') + 2] == '.' || creator.-
field[number - 1][int(letter) - int('A') + 2] == 'm' || creator.field[number - 1][int(letter) -
int('A') + 2] == 'w') &&
                        (creator.field[number + 1][int(letter) - int('A') + 1] == '.' || creator.-
field[number + 1][int(letter) - int('A') + 1] == 'm' || creator.field[number + 1][int(letter) -
int('A') + 1] == 'w') &&
                        (creator.field[number + 1][int(letter) - int('A') + 2] == '.' || creator.-
field[number + 1][int(letter) - int('A') + 2] == 'm' || creator.field[number + 1][int(letter) -
int('A') + 2] == 'w')) {
```

```
                            creator.field[number][int(letter) - int('A') + 1] = 'w';
                            connector.turn = true;
                            creator.turn = false;
                            if (WonGame(creator.field)) {
                                string player_message = to + _MSG_SEP + client_commands[1] +
_MSG_SEP + "youwon" + _MSG_SEP;
                                sprintf(mapped_file.data, "%s", player_message.c_str());
                                cout << "Sending to connector next message:" << player_message <<
'\n';
                                creator.ErasePlayer();
                                connector.ErasePlayer();
                                PrepareField(creator.field);
                                PrepareField(connector.field);
                                game.EraseGame();


                            }
                            else {
                                string player_message = to + _MSG_SEP + client_commands[1] +
_MSG_SEP + "youkilled" + _MSG_SEP;
                                sprintf(mapped_file.data, "%s", player_message.c_str());
                                cout << "Sending to client next message:" << player_message << '\
n';
                            }
                        }
                        else if (creator.field[number][int(letter) - int('A') + 1] == 'w' || cre-
ator.field[number][int(letter) - int('A') + 1] == 'm') {
                            connector.turn = true;
                            creator.turn = false;
                            string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP
+ "yourepeated" + _MSG_SEP;
                            sprintf(mapped_file.data, "%s", player_message.c_str());
                            cout << "Sending to client next message:" << player_message << '\n';
                        }
                        else if (creator.field[number][int(letter) - int('A') + 1] == 'X' &&
                        creator.field[number][int(letter) - int('A') + 2] == 'X' &&
                        (creator.field[number - 1][int(letter) - int('A') + 1] == '.' || creator.-
field[number - 1][int(letter) - int('A') + 1] == 'm' || creator.field[number - 1][int(letter) -
int('A') + 1] == 'w') &&
                        (creator.field[number - 1][int(letter) - int('A') + 2] == '.' || creator.-
field[number - 1][int(letter) - int('A') + 2] == 'm' || creator.field[number - 1][int(letter) -
int('A') + 2] == 'w') &&
                        (creator.field[number + 1][int(letter) - int('A') + 1] == '.' || creator.-
field[number + 1][int(letter) - int('A') + 1] == 'm' || creator.field[number + 1][int(letter) -
int('A') + 1] == 'w') &&
                        (creator.field[number + 1][int(letter) - int('A') + 2] == '.' || creator.-
field[number + 1][int(letter) - int('A') + 2] == 'm' || creator.field[number + 1][int(letter) -
int('A') + 2] == 'w')) {
                            creator.field[number][int(letter) - int('A') + 1] = 'w';
                            connector.turn = true;
                            creator.turn = false;
                            string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP
+ "youwounded" + _MSG_SEP;
                            sprintf(mapped_file.data, "%s", player_message.c_str());
                            cout << "Sending to client next message: " << player_message << '\n';
                        }
                        else if (creator.field[number][int(letter) - int('A') + 1] == 'X' && (cre-
ator.field[number][int(letter) - int('A') + 2] == '.' || creator.field[number][int(letter) -
int('A') + 2] == 'm' || creator.field[number][int(letter) - int('A') + 2] == 'w') &&
                        creator.field[number - 1][int(letter) - int('A') + 1] == 'X' &&
                        (creator.field[number - 1][int(letter) - int('A') + 2] == '.' || creator.-
field[number - 1][int(letter) - int('A') + 2] == 'm' || creator.field[number - 1][int(letter) -
int('A') + 2] == 'w') &&
                        (creator.field[number + 1][int(letter) - int('A') + 1] == '.' || creator.-
field[number + 1][int(letter) - int('A') + 1] == 'm' || creator.field[number + 1][int(letter) -
int('A') + 1] == 'w') &&
                        (creator.field[number + 1][int(letter) - int('A') + 2] == '.' || creator.-
field[number + 1][int(letter) - int('A') + 2] == 'm' || creator.field[number + 1][int(letter) -
int('A') + 2] == 'w')) {
```

```
                            creator.field[number][int(letter) - int('A') + 1] = 'w';
                            connector.turn = true;
                            creator.turn = false;
                            string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP
+ "youwounded" + _MSG_SEP;
                            sprintf(mapped_file.data, "%s", player_message.c_str());
                            cout << "Sending to client next message: " << player_message << '\n';
                        }
                        else if (creator.field[number][int(letter) - int('A') + 1] == 'X' &&
                        (creator.field[number][int(letter) - int('A') + 2] == '.' || creator.-
field[number][int(letter) - int('A') + 2] == 'm' || creator.field[number][int(letter) - int('A') +
2] == 'w') &&
                        (creator.field[number - 1][int(letter) - int('A') + 1] == '.' || creator.-
field[number - 1][int(letter) - int('A') + 1] == 'm' || creator.field[number - 1][int(letter) -
int('A') + 1] == 'w') &&
                        (creator.field[number - 1][int(letter) - int('A') + 2] == '.' || creator.-
field[number - 1][int(letter) - int('A') + 2] == 'm' || creator.field[number - 1][int(letter) -
int('A') + 2] == 'w') &&
                        creator.field[number + 1][int(letter) - int('A') + 1] == 'X' &&
                        (creator.field[number + 1][int(letter) - int('A') + 2] == '.' || creator.-
field[number + 1][int(letter) - int('A') + 2] == 'm' || creator.field[number + 1][int(letter) -
int('A') + 2] == 'w')) {
                            creator.field[number][int(letter) - int('A') + 1] = 'w';
                            connector.turn = true;
                            creator.turn = false;
                            string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP
+ "youwounded" + _MSG_SEP;
                            sprintf(mapped_file.data, "%s", player_message.c_str());
                            cout << "Sending to client next message: " << player_message << '\n';
                        }
                        else if (creator.field[number][int(letter) - int('A') + 1] == 'X' && cre-
ator.field[number + 1][int(letter) - int('A') + 1] == 'X') {
                            creator.field[number][int(letter) - int('A') + 1] = 'w';

                            connector.turn = true;
                            creator.turn = false;
                            string player_message = to + _MSG_SEP + client_commands[1] +_MSG_SEP +
"youwounded" + _MSG_SEP;
                            sprintf(mapped_file.data, "%s", player_message.c_str());
                            cout << "Sending to client next message: " << player_message << '\n';
                        }
                        else if (creator.field[number][int(letter) - int('A') + 1] == '.') {

                            connector.turn = false;
                            creator.turn = true;
                            creator.field[number][int(letter) - int('A') + 1] = 'm';
                            string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP
+ "youmissed" + _MSG_SEP;
                            sprintf(mapped_file.data, "%s", player_message.c_str());
                            cout << "Sending to client next message: " << player_message << '\n';
                        }
                        cout << "Current state of " << creator.username << "'s field is: " << '\
n';
                        PrintField(creator.field);
                    }
                    else {
                        string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"gamenotexists" + _MSG_SEP;
                        sprintf(mapped_file.data, "%s", player_message.c_str());
                        cout << "Sending to client next message: " << player_message << '\n';
                    }
                }
                else {
                    string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP + "noty-
ourturn" + _MSG_SEP;
                    sprintf(mapped_file.data, "%s", player_message.c_str());
                    cout << "Sending to client next message: " << player_message << '\n';
                }
```

```
            }
            // shoot creator
            else if (client_commands[1] == creator.username) {
                if (creator.turn && !connector.turn) {
                    if (game.name == client_commands[3]) {
                        int number = stoi(client_commands[5]);
                        string l = client_commands[4];
                        char letter = l[0];
                        // wounded
                        if (connector.field[number][int(letter) - int('A') + 1] == 'X' &&
                        (connector.field[number][int(letter) - int('A') + 2] == '.' || connector.-
field[number][int(letter) - int('A') + 2] == 'm' || connector.field[number][int(letter) - int('A')
+ 2] == 'w') &&
                        (connector.field[number - 1][int(letter) - int('A') + 1] == '.' || connec-
tor.field[number - 1][int(letter) - int('A') + 1] == 'm' || connector.field[number - 1][int(let-
ter) - int('A') + 1] == 'w') &&
                        (connector.field[number - 1][int(letter) - int('A') + 2] == '.' || connec-
tor.field[number - 1][int(letter) - int('A') + 2] == 'm' || connector.field[number - 1][int(let-
ter) - int('A') + 2] == 'w') &&
                        (connector.field[number + 1][int(letter) - int('A') + 1] == '.' || connec-
tor.field[number + 1][int(letter) - int('A') + 1] == 'm' || connector.field[number + 1][int(let-
ter) - int('A') + 1] == 'w') &&
                        (connector.field[number + 1][int(letter) - int('A') + 2] == '.' || connec-
tor.field[number + 1][int(letter) - int('A') + 2] == 'm' || connector.field[number + 1][int(let-
ter) - int('A') + 2] == 'w')) {
                            connector.field[number][int(letter) - int('A') + 1] = 'w';
                            creator.turn = true;
                            connector.turn = false;
                            if (WonGame(connector.field)) {
                                string player_message = to + _MSG_SEP + client_commands[1] +
_MSG_SEP + "youwon" + _MSG_SEP;
                                sprintf(mapped_file.data, "%s", player_message.c_str());
                                cout << "Sending to creator next message: " << player_message <<
'\n';
                                creator.ErasePlayer();
                                connector.ErasePlayer();
                                PrepareField(creator.field);
                                PrepareField(connector.field);
                                game.EraseGame();
                            }
                            else {
                                string player_message = to + _MSG_SEP + client_commands[1] +
_MSG_SEP + "youkilled" + _MSG_SEP;
                                sprintf(mapped_file.data, "%s", player_message.c_str());
                                cout << "Sending to client next message: " << player_message << '\
n';
                            }
                        }
                        else if (connector.field[number][int(letter) - int('A') + 1] == 'w' ||
connector.field[number][int(letter) - int('A') + 1] == 'm') {
                            creator.turn = true;
                            connector.turn = false;
                            string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP
+ "yourepeated" + _MSG_SEP;
                            sprintf(mapped_file.data, "%s", player_message.c_str());
                            cout << "Sending to client next message: " << player_message << '\n';
                        }
                        else if (connector.field[number][int(letter) - int('A') + 1] == 'X' &&
                        connector.field[number][int(letter) - int('A') + 2] == 'X' &&
                        (connector.field[number - 1][int(letter) - int('A') + 1] == '.' || connec-
tor.field[number - 1][int(letter) - int('A') + 1] == 'm' || connector.field[number - 1][int(let-
ter) - int('A') + 1] == 'w') &&
                        (connector.field[number - 1][int(letter) - int('A') + 2] == '.' || connec-
tor.field[number - 1][int(letter) - int('A') + 2] == 'm' || connector.field[number - 1][int(let-
ter) - int('A') + 2] == 'w') &&
                        (connector.field[number + 1][int(letter) - int('A') + 1] == '.' || connec-
tor.field[number + 1][int(letter) - int('A') + 1] == 'm' || connector.field[number + 1][int(let-
ter) - int('A') + 1] == 'w') &&
                        (connector.field[number + 1][int(letter) - int('A') + 2] == '.' || connec-
tor.field[number + 1][int(letter) - int('A') + 2] == 'm' || connector.field[number + 1][int(let-
ter) - int('A') + 2] == 'w')) {
                            connector.field[number][int(letter) - int('A') + 1] = 'w';
                            creator.turn = true;
                            connector.turn = false;
                            string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP
+ "youwounded" + _MSG_SEP;
                            sprintf(mapped_file.data, "%s", player_message.c_str());
                            cout << "Sending to client next message: " << player_message << '\n';
                        }
                        else if (connector.field[number][int(letter) - int('A') + 1] == 'X' &&
(connector.field[number][int(letter) - int('A') + 2] == '.' || connector.field[number][int(letter)
- int('A') + 2] == 'm' || connector.field[number][int(letter) - int('A') + 2] == 'w') &&
                        connector.field[number - 1][int(letter) - int('A') + 1] == 'X' &&
                        (connector.field[number - 1][int(letter) - int('A') + 2] == '.' || connec-
tor.field[number - 1][int(letter) - int('A') + 2] == 'm' || connector.field[number - 1][int(let-
ter) - int('A') + 2] == 'w') &&
                        (connector.field[number + 1][int(letter) - int('A') + 1] == '.' || connec-
tor.field[number + 1][int(letter) - int('A') + 1] == 'm' || connector.field[number + 1][int(let-
ter) - int('A') + 1] == 'w') &&
                        (connector.field[number + 1][int(letter) - int('A') + 2] == '.' || connec-
tor.field[number + 1][int(letter) - int('A') + 2] == 'm' || connector.field[number + 1][int(let-
ter) - int('A') + 2] == 'w')) {
                            connector.field[number][int(letter) - int('A') + 1] = 'w';
                            creator.turn = true;
                            connector.turn = false;
                            string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP
+ "youwounded" + _MSG_SEP;
                            sprintf(mapped_file.data, "%s", player_message.c_str());
                            cout << "Sending to client next message: " << player_message << '\n';
                        }
                        else if (connector.field[number][int(letter) - int('A') + 1] == 'X' &&
                        (connector.field[number][int(letter) - int('A') + 2] == '.' || connector.-
field[number][int(letter) - int('A') + 2] == 'm' || connector.field[number][int(letter) - int('A')
+ 2] == 'w') &&
                        (connector.field[number - 1][int(letter) - int('A') + 1] == '.' || connec-
tor.field[number - 1][int(letter) - int('A') + 1] == 'm' || connector.field[number - 1][int(let-
ter) - int('A') + 1] == 'w') &&
                        (connector.field[number - 1][int(letter) - int('A') + 2] == '.' || connec-
tor.field[number - 1][int(letter) - int('A') + 2] == 'm' || connector.field[number - 1][int(let-
ter) - int('A') + 2] == 'w') &&
                        connector.field[number + 1][int(letter) - int('A') + 1] == 'X' &&
                        (connector.field[number + 1][int(letter) - int('A') + 2] == '.' || connec-
tor.field[number + 1][int(letter) - int('A') + 2] == 'm' || connector.field[number + 1][int(let-
ter) - int('A') + 2] == 'w')) {
                            connector.field[number][int(letter) - int('A') + 1] = 'w';
                            creator.turn = true;
                            connector.turn = false;
                            string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP
+ "youwounded" + _MSG_SEP;
                            sprintf(mapped_file.data, "%s", player_message.c_str());
                            cout << "Sending to client next message: " << player_message << '\n';
                        }
                        else if (connector.field[number][int(letter) - int('A') + 1] == 'X' &&
connector.field[number + 1][int(letter) - int('A') + 1] == 'X') {
                            connector.field[number][int(letter) - int('A') + 1] = 'w';
                            connector.turn = true;
                            creator.turn = false;
                            string player_message = to + _MSG_SEP + client_commands[1] +_MSG_SEP +
"youwounded" + _MSG_SEP;
                            sprintf(mapped_file.data, "%s", player_message.c_str());
                            cout << "Sending to client next message: " << player_message << '\n';
                        }
                        else if (connector.field[number][int(letter) - int('A') + 1] == '.') {
                            creator.turn = false;
                            connector.turn = true;
                            connector.field[number][int(letter) - int('A') + 1] = 'm';
                            string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP
+ "youmissed" + _MSG_SEP;
                            sprintf(mapped_file.data, "%s", player_message.c_str());
                            cout << "Sending to client next message: " << player_message << '\n';
                        }
                        cout << "Current state of " << connector.username << "'s field is: " << '\
n';
                        PrintField(connector.field);
                    }
                    else {
                        string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"gamenotexists" + _MSG_SEP;
                        sprintf(mapped_file.data, "%s", player_message.c_str());
                        cout << "Sending to client next message: " << player_message << '\n';
                    }
                }
                else {
                    creator.turn = false;
                    string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP + "noty-
ourturn" + _MSG_SEP;
                    sprintf(mapped_file.data, "%s", player_message.c_str());
                    cout << "Sending to client next message: " << player_message << '\n';
                }
            }
        }
        else if(client_commands[2]=="print_self"){
           vector< vector<char>> field;
            if (client_commands[1] == connector.username) {
                field = connector.field;

            }
            else{
                field = creator.field;
            }
            string player_message = to + _MSG_SEP + "print_self" +_MSG_SEP + "  ";

            for (int i = 0; i<10; ++i){
                player_message = player_message + char(int('A')+i) + " ";
            }
            player_message = player_message + _MSG_SEP;
            for (int i = 1; i < 11; ++i) {
                player_message = player_message + to_string(i)  + "  ";
                for (int j = 1; j < 11; ++j) {
                    player_message = player_message + field[i][j] + " ";
                }
                player_message = player_message + _MSG_SEP;
            }
            sprintf(mapped_file.data, "%s", player_message.c_str());
            cout << "Sending to client next message: " << "your field" << '\n';
        }
        else if(client_commands[2]=="print_oppon"){
           vector< vector<char>> field;
            if (client_commands[1] != connector.username) {
                field = connector.field;

            }
            else{
                field = creator.field;
            }
            string player_message = to + _MSG_SEP + "print_oppon" +_MSG_SEP + "  ";

                for (int i = 0; i<10; ++i){
                    player_message = player_message + char(int('A')+i) + " ";
                }
                player_message = player_message + _MSG_SEP;
                for (int i = 1; i < 11; ++i) {
                    player_message = player_message + to_string(i)  + "  ";
                    for (int j = 1; j < 11; ++j) {
                        if(field[i][j] != 'X')
                            player_message = player_message + field[i][j] + " ";
                        else
                            player_message = player_message + "." + " ";
                    }
                    player_message = player_message + _MSG_SEP;
                }
                sprintf(mapped_file.data, "%s", player_message.c_str());
                cout << "Sending to client next message: " << "opponent's field" << '\n';
        }



        else if (client_commands[2] == "disconnect") {
            if (client_commands[1] == creator.username) {
                creator.turn = false;
                connector.turn = true;
                game.connected = false;
                string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP + "discon-
nected" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                cout << "Sending to client next message: " << player_message << std::endl;
            }
            else {
                creator.turn = true;
                connector.turn = false;
                game.connected = false;
                string player_message = to + _MSG_SEP + connector.username + _MSG_SEP + "discon-
nected" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                cout << "Sending to client next message: " << player_message << '\n';
            }
        }

        pthread_mutex_unlock(&mutex);
        cout << "Unlocked mutex" << '\n';


    }

    if (er = pthread_mutex_destroy(&mutex))
    {
        printf("Mutex destroy error: %d", er);
        return -1;
    }


    if (shm_unlink(_BUFFER_NAME) == -1) {
        perror("An error while unlink mutex has been detected!\n");
        return -1;
    }
    return 0;
}
```

client.cpp

```
#include <iostream>
#include <fcntl.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/mman.h>
#include <cassert>
```

```
#include <cstring>
#include <vector>
#include "MappedFile.hpp"
#include "Player_Game.hpp"
#include <algorithm>
#include <sys/stat.h>
#include <fstream>

using namespace std;

MappedFile mapped_file;
pthread_mutex_t mutex;
string nickname;
string username, password;
bool playing = false;
string current_game = "";

void SendMessage (const string &message) {
    if (pthread_mutex_lock(&mutex) != 0) {
        cout << "An error while locking mutex has been detected!" << '\n';
        exit(EXIT_FAILURE);
    }
    memset(mapped_file.data, '\0', _MAPPED_SIZE);
    sprintf(mapped_file.data, "%s", message.c_str());
    pthread_mutex_unlock(&mutex);
}

bool ReceiveAnswer() {
    if (mapped_file.data[0] != 'T' || mapped_file.data[1] != 'O' || mapped_file.data[2] != _MSG_SEP)
{
        return false;
    }
    string message = mapped_file.data;
    vector<string> server_commands;
    string strings = "";
    // считывание из мапы

    for (int i = 0; i < message.size(); i++) {
        if (message[i] == _MSG_SEP) {
            server_commands.push_back(strings);
            strings = "";
        }
        else {
            strings.push_back(message[i]);
        }
    }

    if(server_commands[1] == "print_self"){
        for (int i=2; i<server_commands.size();++i){
            cout << server_commands[i] << '\n';
        }
        return true;
    }
    else if(server_commands[1] == "print_oppon"){
        for (int i=2; i<server_commands.size();++i){
            cout << server_commands[i] << '\n';
        }
        return true;
    }

    else if (server_commands[1] == nickname) {
        if (pthread_mutex_lock(&mutex) != 0) {
            perror("Error locking mutex\n");
            return -1;
        }
        memset(mapped_file.data, '\0', _MAPPED_SIZE);
        // pthread_mutex_unlock(mutex.ptr);
        if (server_commands[2] == "gamecreated") {
            playing = true;
            cout << "Created successfully!" << '\n';
            cout << "You are a player №1, cause you have created the game. Your field has been pre-
pared!" << '\n';
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "connected") {
            cout << "Connected sucessfully" << '\n';
            cout << "You are a player №2, cause you have connected to the game. Your field has been
prepared!" << '\n';
            playing = true;
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "checked") {
            cout << "Connected sucessfully" << '\n';
            cout << "You are a player №2, cause you have connected to the game. Your field has been
prepared!" << '\n';
            current_game = server_commands[3];
            password = server_commands[4];
            playing = true;
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "notchecked") {
            cout << "Connected not sucessfully" << '\n';
            pthread_mutex_unlock(&mutex);
            return true;
        }

        if (server_commands[2] == "notatgame") {
            playing = true;
            cout << "You can't play without another player!" << '\n';
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "gamenotexists") {
            cout << "Game with this name not exists" << '\n';
            playing = false;
            current_game = "";
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "wrongpassword") {
            cout << "Wrong password has been detected!" << '\n';
            playing = false;
            current_game = "";
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "notyourturn") {
            cout << "It's not your turn now!" << '\n';
            playing = true;
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "youwounded") {
            playing = true;
            cout << "You have wounded enemy's ship! Please enter coordinates again!" << '\n';
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "youmissed") {
            playing = true;
            cout << "Unfortunately you have missed! Now it's your enemy's turn!" << '\n';
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "youkilled") {
            playing = true;
            cout << "Congrats, you have KILLED enemy's ship! Please enter coordinates again!" << '\
n';
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "zeroplaces") {
            playing = false;
            cout << "Sorry, but you can not create a game or connect to existing game. There are not
free places!" << '\n';
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "yourepeated") {
            playing = true;
            cout << "You have already entered these coordinates! Please enter something new." << '\
n';
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "disconnected") {
            cout << "You have successfully disconnected from the server!" << '\n';
            playing = false;
            pthread_mutex_unlock(&mutex);
            return true;
        }
        if (server_commands[2] == "youwon") {
            cout << "YOU WON THE GAME!" << '\n';
            playing = false;
            pthread_mutex_unlock(&mutex);
            return true;
        }
        else {
            cout  << "Warning: unknown message has been detected!" << '\n';
            playing = false;
            pthread_mutex_unlock(&mutex);
            return true;
        }
        pthread_mutex_unlock(&mutex);
        return true;

    }
    else if (server_commands[1] == username)
    {
        if(server_commands[2] == "invited"){
            cout << "Invited successfully!\n";
            pthread_mutex_unlock(&mutex);
            return true;
        }
    }

    return false;
}

void Help() {
    cout << "Follow next rules: " << '\n';
    cout << '\t' << "create for creating a new game" << '\n';
    cout << '\t' << "connect for connecting to the server" << '\n';
    cout << '\t' << "shoot for shooting at enemy's ship" << '\n';
    cout << '\t' << "print for checking your field" << '\n';
    cout << '\t' << "invite for sending invite to opponent" << '\n';
    cout << '\t' << "check invite" << '\n';
    cout << '\t' << "disconnect for leaving from the server" << '\n';
    cout << '\t' << "quit for leaving from the program" << '\n';
    cout << '\t' << "help for checking rules" << '\n';
```

```
}

int main() {
    mapped_file.fd = shm_open(_BUFFER_NAME, O_RDWR, _SHM_OPEN_MODE);
    if (mapped_file.fd == -1 ) {
        perror("An error while shm_open has been detected!\n");
        return -1;
    }
    int er;
    if (er = pthread_mutex_init(&mutex, NULL))
    {
        printf("Mutex init error: %d", er);
        return -1;
    }
    mapped_file.data = (char*)mmap(0, _MAPPED_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED,
mapped_file.fd, 0);
    if (mapped_file.data == MAP_FAILED) {
        perror("An error while mmaping has been detected!\n");
    }
    cout << "Welcome to the SeaBattle! Please enter your nickname: " << '\n';
    cout << "> ";
    cin >> nickname;
    cout << "Hello, " << nickname << "!\n";
    Help();
    string command;
    string gamename;
    while (cout << "> " && cin >> command) {
        if (!playing && command == "create") {
            cin >> gamename >> password;
            current_game = gamename;
            string server_message = on + _MSG_SEP + nickname + _MSG_SEP + "create" + _MSG_SEP +
gamename + _MSG_SEP + password + _MSG_SEP;
            SendMessage (server_message);
            bool hasnotanswer = true;
            while (hasnotanswer) {
                hasnotanswer = !ReceiveAnswer();
            }
        }

        else if (playing && command == "create") {
            cin >> gamename >> password;
            cout << "Can't create a new game, you are playing now! Please enter another command!" <<
'\n';
            continue;
        }
        else if (!playing && command == "connect") {
            cin >> gamename >> password;
            current_game = gamename;
            string server_message = on + _MSG_SEP + nickname + _MSG_SEP + "connect" + _MSG_SEP +
gamename + _MSG_SEP + password + _MSG_SEP;
            SendMessage (server_message);
            bool hasnotanswer = true;
            while (hasnotanswer) {
                hasnotanswer = !ReceiveAnswer();
            }
        }
        else if (playing && command == "connect") {
            cin >> gamename >> password;
            cout << "Can't connect to a new game, you've already connected! Please enter another
command!" << '\n';
            continue;
        }
        else if (playing && command == "print") {
            string server_message = on + _MSG_SEP + nickname + _MSG_SEP + "print_self" + _MSG_SEP;
            SendMessage (server_message);
            cout << "Your field!\n";
            bool hasnotanswer = true;
            while (hasnotanswer) {
                hasnotanswer = !ReceiveAnswer();
            }
            server_message = on + _MSG_SEP + nickname + _MSG_SEP + "print_oppon" + _MSG_SEP;
            SendMessage (server_message);
            cout << "\nOpponent's field!\n";
            hasnotanswer = true;
            while (hasnotanswer) {
                hasnotanswer = !ReceiveAnswer();
            }
        }
        else if (playing && command == "shoot") {
            int number;
                    char letter;
            cin >> letter >> number;
            if ((!((letter >= 'A') && (letter <= 'J'))) || ((number < 1) || (number > 10))) {
                cout << "Please enter letter between A and J and number between 1 and 10!" << '\n';
                continue;
            }
            else {

                string server_message = on + _MSG_SEP + nickname + _MSG_SEP + "shoot" + _MSG_SEP +
current_game + _MSG_SEP + letter + _MSG_SEP + to_string(number) + _MSG_SEP;
                SendMessage (server_message);
                bool hasnotanswer = true;
                while (hasnotanswer) {
                    hasnotanswer = !ReceiveAnswer();
                }
            }
        }
        else if (playing && command == "invite") {

            cin >> username;
            string server_message = on + _MSG_SEP + username + _MSG_SEP + "invite" + _MSG_SEP +
gamename + _MSG_SEP + password + _MSG_SEP;
            SendMessage (server_message);
            bool hasnotanswer = true;
            while (hasnotanswer) {
                hasnotanswer = !ReceiveAnswer();
            }
        }
        else if (!playing && command == "check") {
            string server_message = on + _MSG_SEP + nickname + _MSG_SEP + "check" + _MSG_SEP;
            SendMessage (server_message);
            bool hasnotanswer = true;
            while (hasnotanswer) {
                hasnotanswer = !ReceiveAnswer();
            }
        }
        else if (!playing && command == "shoot") {
            int number;
            char letter;
            cin >> letter >> number;
            cout << "You are not in the game right now. Please create a game or connect to the ex-
isting one!" << '\n';
            continue;
        }
        else if (playing && command == "disconnect") {
            string server_message = on + _MSG_SEP + nickname + _MSG_SEP + "disconnect" + _MSG_SEP +
current_game + _MSG_SEP;
            SendMessage (server_message);
            bool hasnotanswer = true;
            while (hasnotanswer) {
                hasnotanswer = !ReceiveAnswer();
            }
        }
        else if (command == "help") {
            Help();
        }
        else if (!playing && command == "quit") {
            break;
        }
        else {
            cout << "Wrong input!" << '\n';
        }
    }
    if (er = pthread_mutex_destroy(&mutex))
    {
        printf("Mutex destroy error: %d", er);
        return -1;
    }
    return 0;
}
```

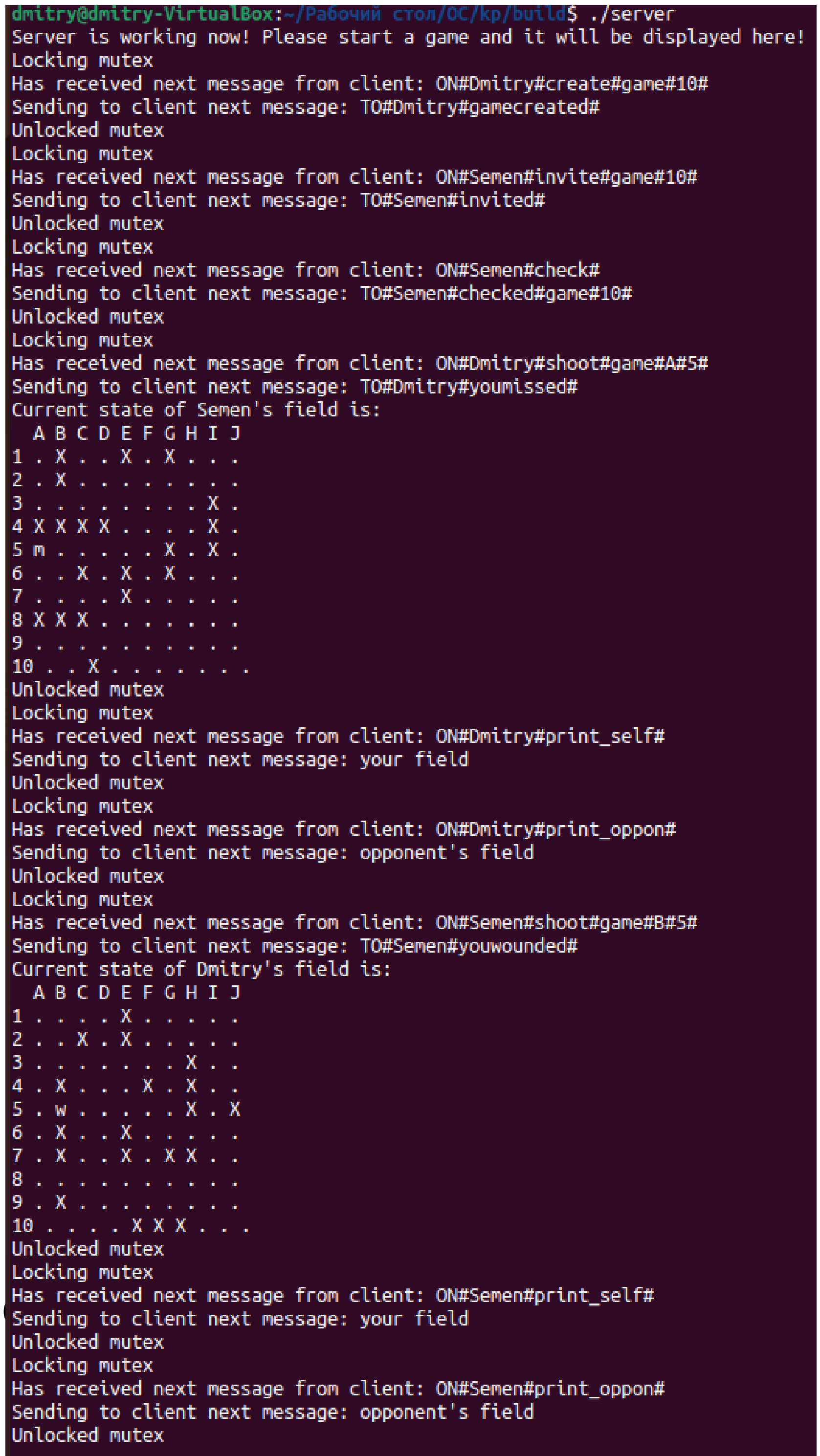
MappedFile.hpp

```
#ifndef MAPPED_FILE_HPP
#define MAPPED_FILE_HPP

// constants
#define _MAPPED_SIZE 8192
#define _SHM_OPEN_MODE S_IWUSR | S_IRUSR | S_IRGRP | S_IROTH
#define _BUFFER_NAME "buffer"
#define _MSG_SEP '#'

std:: string on = "ON";
std:: string to = "TO";

struct MappedFile {
    int fd;
    char *data;
};

#endif
```

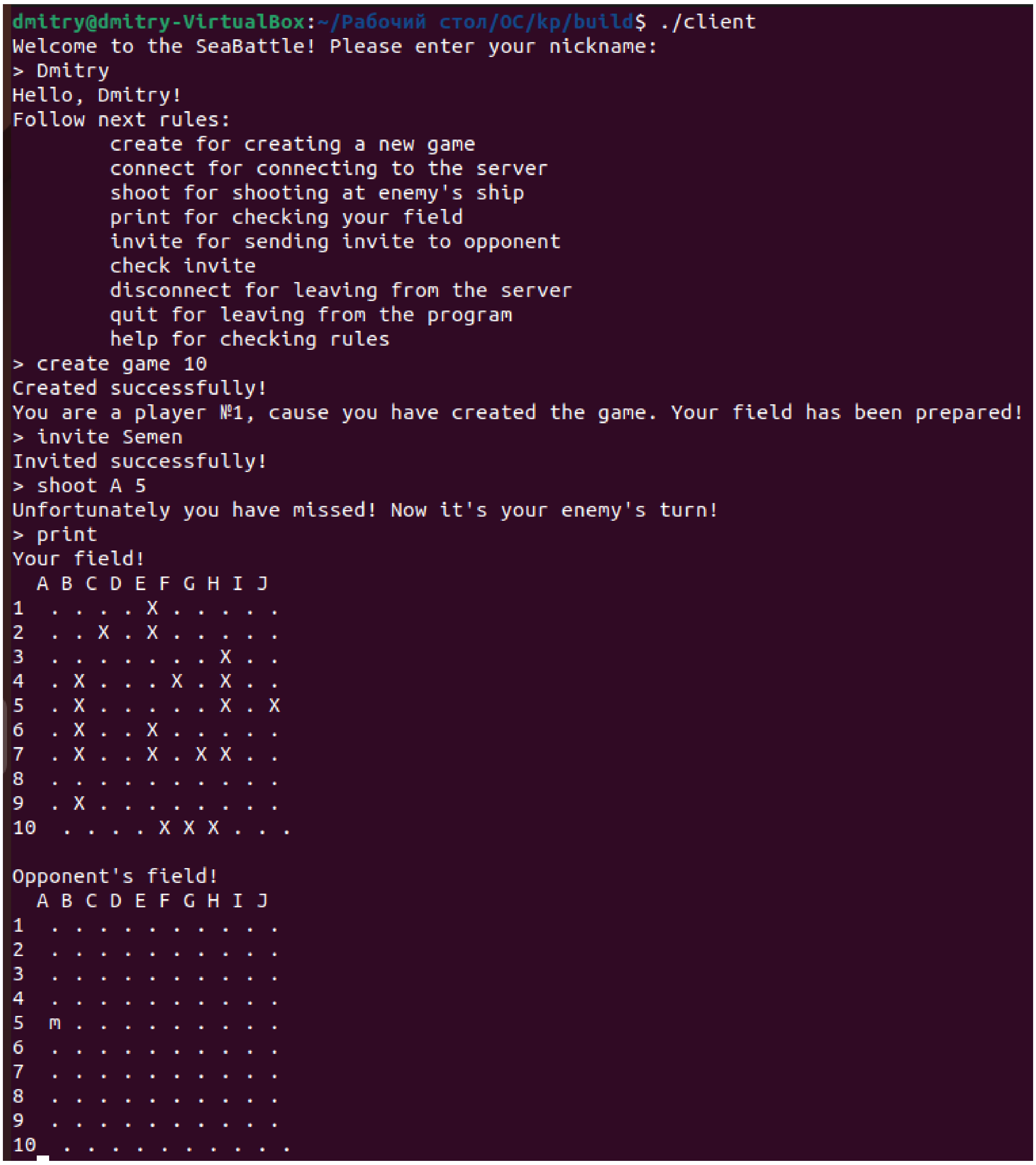
Player_Game.hpp

```
#ifndef PLAYERANDGAME_H
#define PLAYERANDGAME_H
#include <algorithm>
#include <vector>
#include <string>
#include <ctime>
#include <iostream>

using namespace std;


class Player {
    public:
        string username;
        vector<vector<char>> field;
        bool turn;
        bool invite;
        Player() : field(12,  vector<char> (12, '.')), username(""), turn(false), invite(false) {}
        void ErasePlayer() {
            username = "";
            turn = false;
            invite = false;
        }
};
class Game {
    public:
        string name;
        string password;
        bool connected;
        bool created;
        Game() : name(""), password(""), connected(false), created(false) {}
        void EraseGame() {
            name = "";
            password = "";
            connected = false;
            created = false;
        }
};
void Map ( vector< vector<char>> &field) {
    int j =- 1, k, v, l, x[2], y;
    srand(time(0));
    for (l = 4; l > 0; l--) {
        for (k = 5; k - l; k--) {
            v = 1&rand();
            do for (x[v] = 1 + rand() % 10, x[1 - v] = 1 + rand() % 7, y = j = 0; j - l; y |= field[x[0]]
[x[1]] != '.', x[1 - v]++, j++); while(y);
            x[1 - v] -= l + 1, field[x[0]][x[1]] = '/', x[v]--, field[x[0]][x[1]] = '/', x [v] += 2, field[x[0]]
[x[1]] = '/', x[v]--, x[1 - v]++;
            for (j = -1; ++j - l; field[x[0]][x[1]] = 'X', x[v]--, field[x[0]][x[1]] = '/', x[v] += 2, field[x[0]]
[x[1]] = '/', x[v]--, x[1 - v]++);
            field[x[0]][x[1]] = '/', x[v]--, field[x[0]][x[1]] = '/', x[v]+=2, field[x[0]][x[1]] = '/';
        }
    }
    for (int i = 0; i < 12; ++i) {
         replace(field[i].begin(), field[i].end(), '/', '.');
    }
}
void PrintField ( vector< vector<char>> &field) {
    cout << "  ";
    for (int i = 0; i<10; ++i){
        cout << char(int('A')+i) << " ";
    }
    cout << '\n';
    for (int i = 1; i < 11; ++i) {
        cout << i << " ";
        for (int j = 1; j < 11; ++j) {
            cout << field[i][j] << " ";
        }
        cout <<  '\n';
    }
}
bool WonGame ( vector< vector<char>> &field) {
    for (int i = 1; i < 11; ++i) {
        for (int j = 1; j < 11; ++j) {
            if (field[i][j] == 'X') {
                return false;
            }
        }
    }
    return true;
}
void PrepareField ( vector< vector<char>>& field) {
        for (int i = 0; i < 12; i++) {
                field[i].clear();
                field[i] =  vector<char>(12, '.');
        }
}
#endif
```

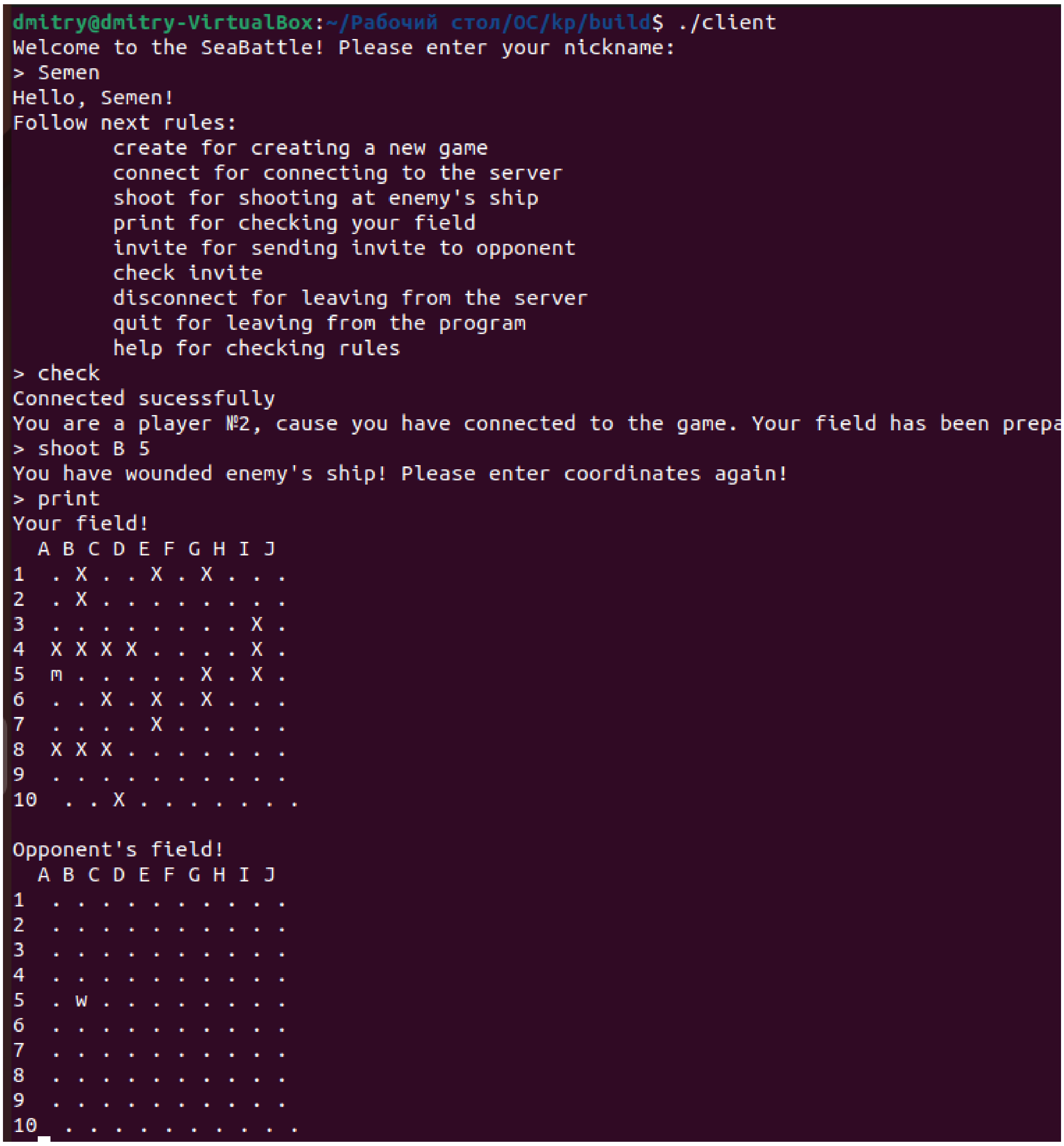
Демонстрация работы программы



```
dmitry@dmitry-VirtualBox:~/Рабочий стол/ОС/kp/build$ ./server
Server is working now! Please start a game and it will be displayed here!
Locking mutex
Has received next message from client: ON#Dmitry#create#game#10#
Sending to client next message: TO#Dmitry#gamecreated#
Unlocked mutex
Locking mutex
Has received next message from client: ON#Semen#invite#game#10#
Sending to client next message: TO#Semen#invited#
Unlocked mutex
Locking mutex
Has received next message from client: ON#Semen#check#
Sending to client next message: TO#Semen#checked#game#10#
Unlocked mutex
Locking mutex
Has received next message from client: ON#Dmitry#shoot#game#A#5#
Sending to client next message: TO#Dmitry#youmissed#
Current state of Semen's field is:
  A B C D E F G H I J
1 . X . . X . X . . .
2 . X . . . . . . . .
3 . . . . . . . . X .
4 X X X X . . . . X .
5 m . . . . . X . X .
6 . . X . X . X . . .
7 . . . . X . . . . .
8 X X X . . . . . . .
9 . . . . . . . . . .
10 . . X . . . . . . .
Unlocked mutex
Locking mutex
Has received next message from client: ON#Dmitry#print_self#
Sending to client next message: your field
Unlocked mutex
Locking mutex
Has received next message from client: ON#Dmitry#print_oppon#
Sending to client next message: opponent's field
Unlocked mutex
Locking mutex
Has received next message from client: ON#Semen#shoot#game#B#5#
Sending to client next message: TO#Semen#youwounded#
Current state of Dmitry's field is:
  A B C D E F G H I J
1 . . . . X . . . . .
2 . . X . X . . . . .
3 . . . . . . . X . .
4 . X . . . X . X . .
5 . w . . . . . X . X
6 . X . . X . . . . .
7 . X . . X . X X . .
8 . . . . . . . . . .
9 . X . . . . . . . .
10 . . . . X X X . . .
Unlocked mutex
Locking mutex
Has received next message from client: ON#Semen#print_self#
Sending to client next message: your field
Unlocked mutex
Locking mutex
Has received next message from client: ON#Semen#print_oppon#
Sending to client next message: opponent's field
Unlocked mutex
```



```
dmitry@dmitry-VirtualBox:~/Рабочий стол/OC/kp/build$ ./client
Welcome to the SeaBattle! Please enter your nickname:
> Dmitry
Hello, Dmitry!
Follow next rules:
        create for creating a new game
        connect for connecting to the server
        shoot for shooting at enemy's ship
        print for checking your field
        invite for sending invite to opponent
        check invite
        disconnect for leaving from the server
        quit for leaving from the program
        help for checking rules
> create game 10
Created successfully!
You are a player №1, cause you have created the game. Your field has been prepared!
> invite Semen
Invited successfully!
> shoot A 5
Unfortunately you have missed! Now it's your enemy's turn!
> print
Your field!
  A B C D E F G H I J
1  . . . . X . . . . .
2  . . X . X . . . . .
3  . . . . . . . X . .
4  . X . . . X . X . .
5  . X . . . . . X . X
6  . X . . X . . . . .
7  . X . . X . X X . .
8  . . . . . . . . . .
9  . X . . . . . . . .
10  . . . . X X X . . .

Opponent's field!
  A B C D E F G H I J
1  . . . . . . . . . .
2  . . . . . . . . . .
3  . . . . . . . . . .
4  . . . . . . . . . .
5  m . . . . . . . . .
6  . . . . . . . . . .
7  . . . . . . . . . .
8  . . . . . . . . . .
9  . . . . . . . . . .
10  . . . . . . . . . .
```



```
dmitry@dmitry-VirtualBox:~/Рабочий стол/OC/kp/build$ ./client
Welcome to the SeaBattle! Please enter your nickname:
> Semen
Hello, Semen!
Follow next rules:
        create for creating a new game
        connect for connecting to the server
        shoot for shooting at enemy's ship
        print for checking your field
        invite for sending invite to opponent
        check invite
        disconnect for leaving from the server
        quit for leaving from the program
        help for checking rules
> check
Connected sucessfully
You are a player №2, cause you have connected to the game. Your field has been prepa
> shoot B 5
You have wounded enemy's ship! Please enter coordinates again!
> print
Your field!
  A B C D E F G H I J
1  . X . . X . X . . .
2  . X . . . . . . . .
3  . . . . . . . . X .
4  X X X X . . . . X .
5  m . . . . . X . X .
6  . . X . X . X . . .
7  . . . . X . . . . .
8  X X X . . . . . . .
9  . . . . . . . . . .
10  . . X . . . . . . .

Opponent's field!
  A B C D E F G H I J
1  . . . . . . . . . .
2  . . . . . . . . . .
3  . . . . . . . . . .
4  . . . . . . . . . .
5  . w . . . . . . . .
6  . . . . . . . . . .
7  . . . . . . . . . .
8  . . . . . . . . . .
9  . . . . . . . . . .
10  . . . . . . . . . .
```

Выводы

Составлена и отлажена программа на языке С++, реализующая консольно-серверную игру. Общение между пользователями и сервером происходит по средством memory map. В системе реализована игра морской бой.