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**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

[https://github.com/](https://github.com/Liguha/OS)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**

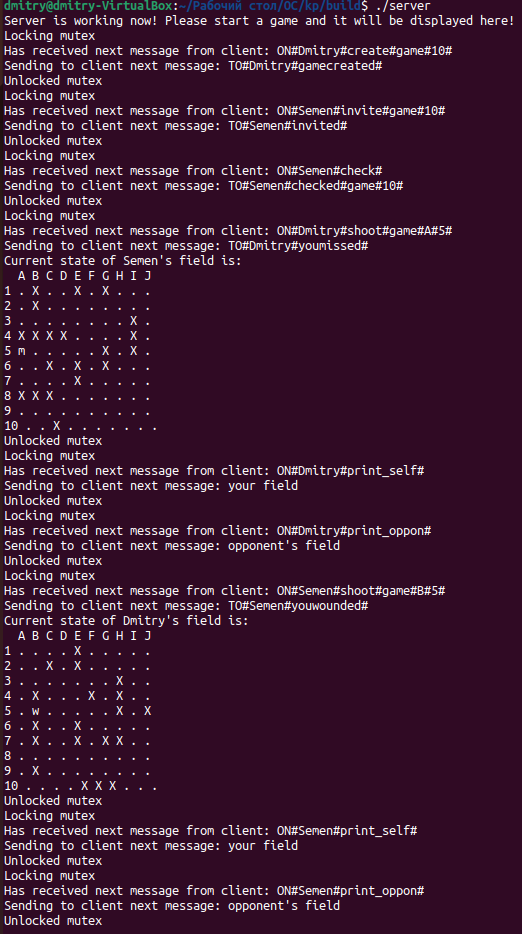
|  |
| --- |
| **server.cpp** |
| #include <fcntl.h>  #include <pthread.h>  #include <sys/mman.h>  #include <sys/stat.h>  #include <unistd.h>  #include <cassert>  #include <cstring>  #include <iostream>  #include <map>  #include <vector>  #include "MappedFile.hpp"  #include "Player\_Game.hpp"  #include <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 + "zeroplaces" + \_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 + "gamecreated" + \_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 + "zeroplaces" + \_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 + "zeroplaces" + \_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 + "notatgame" + \_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' || creator.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' && (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] == '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' && creator.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 + "notyourturn" + \_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] == '.' || connector.field[number - 1][int(letter) - int('A') + 1] == 'm' || connector.field[number - 1][int(letter) - int('A') + 1] == 'w') &&  (connector.field[number - 1][int(letter) - int('A') + 2] == '.' || connector.field[number - 1][int(letter) - int('A') + 2] == 'm' || connector.field[number - 1][int(letter) - int('A') + 2] == 'w') &&  (connector.field[number + 1][int(letter) - int('A') + 1] == '.' || connector.field[number + 1][int(letter) - int('A') + 1] == 'm' || connector.field[number + 1][int(letter) - int('A') + 1] == 'w') &&  (connector.field[number + 1][int(letter) - int('A') + 2] == '.' || connector.field[number + 1][int(letter) - int('A') + 2] == 'm' || connector.field[number + 1][int(letter) - 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] == '.' || connector.field[number - 1][int(letter) - int('A') + 1] == 'm' || connector.field[number - 1][int(letter) - int('A') + 1] == 'w') &&  (connector.field[number - 1][int(letter) - int('A') + 2] == '.' || connector.field[number - 1][int(letter) - int('A') + 2] == 'm' || connector.field[number - 1][int(letter) - int('A') + 2] == 'w') &&  (connector.field[number + 1][int(letter) - int('A') + 1] == '.' || connector.field[number + 1][int(letter) - int('A') + 1] == 'm' || connector.field[number + 1][int(letter) - int('A') + 1] == 'w') &&  (connector.field[number + 1][int(letter) - int('A') + 2] == '.' || connector.field[number + 1][int(letter) - int('A') + 2] == 'm' || connector.field[number + 1][int(letter) - 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] == '.' || connector.field[number - 1][int(letter) - int('A') + 2] == 'm' || connector.field[number - 1][int(letter) - int('A') + 2] == 'w') &&  (connector.field[number + 1][int(letter) - int('A') + 1] == '.' || connector.field[number + 1][int(letter) - int('A') + 1] == 'm' || connector.field[number + 1][int(letter) - int('A') + 1] == 'w') &&  (connector.field[number + 1][int(letter) - int('A') + 2] == '.' || connector.field[number + 1][int(letter) - int('A') + 2] == 'm' || connector.field[number + 1][int(letter) - 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] == '.' || connector.field[number - 1][int(letter) - int('A') + 1] == 'm' || connector.field[number - 1][int(letter) - int('A') + 1] == 'w') &&  (connector.field[number - 1][int(letter) - int('A') + 2] == '.' || connector.field[number - 1][int(letter) - int('A') + 2] == 'm' || connector.field[number - 1][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] == '.' || connector.field[number + 1][int(letter) - int('A') + 2] == 'm' || connector.field[number + 1][int(letter) - 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 + "notyourturn" + \_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 + "disconnected" + \_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 + "disconnected" + \_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;  } |

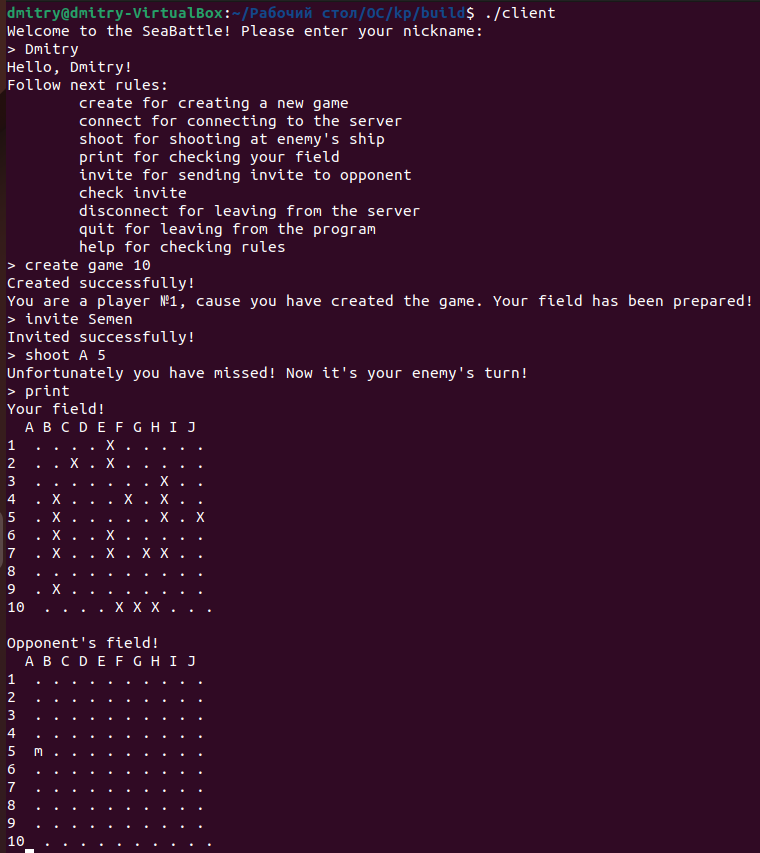
|  |
| --- |
| **client.cpp** |
| #include <iostream>  #include <fcntl.h>  #include <unistd.h>  #include <pthread.h>  #include <sys/mman.h>  #include <cassert>  #include <cstring>  #include <vector>  #include "MappedFile.hpp"  #include "Player\_Game.hpp"  #include <algorithm>  #include <sys/stat.h>  #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 prepared!" << '\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 existing 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;  } |

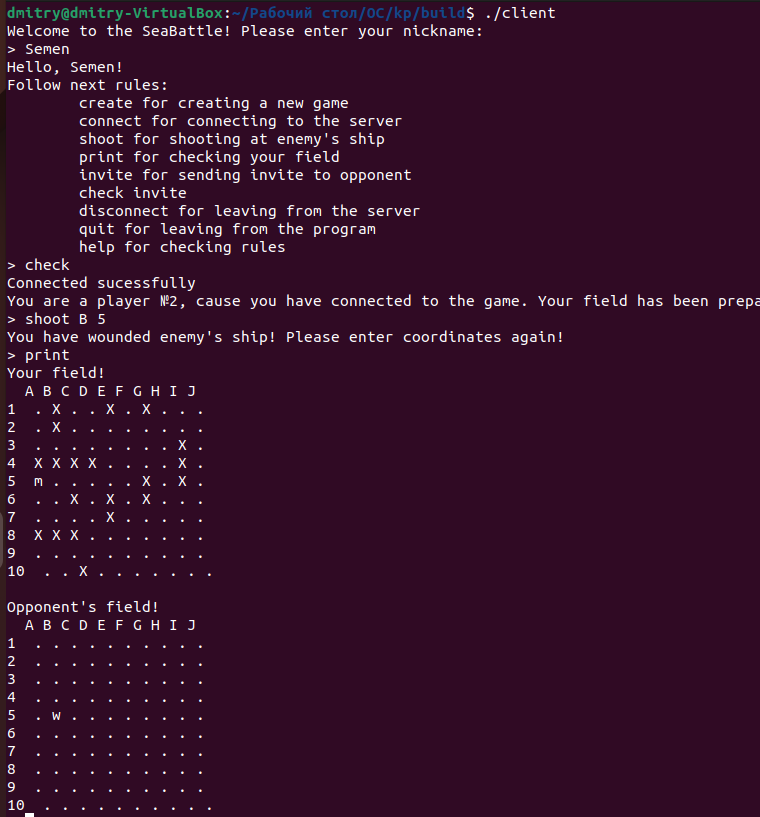
|  |
| --- |
| **MappedFile.hpp** |
| #ifndef MAPPED\_FILE\_HPP  #define MAPPED\_FILE\_HPP  // constants  #define \_MAPPED\_SIZE 8192  #define \_SHM\_OPEN\_MODE S\_IWUSR | S\_IRUSR | S\_IRGRP | S\_IROTH  #define \_BUFFER\_NAME "buffer"  #define \_MSG\_SEP '#'  std:: string on = "ON";  std:: string to = "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**







**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.