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-mapped files”**

Студент: Постнов Александр Вячеславович

Группа: М8О-201Б-21

Вариант: 5

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/kappaprideonly/mai-os-labs

**Постановка задачи**

Морской бой. Общение между сервером и клиентом необходимо организовать при помощи memory map. Каждый игрок должен при запуске ввести свой логин. Для каждого игрока должна вестись статистика игр (сколько побед/поражений). Игрок может посмотреть свою статистику.

**Общие сведения о программе**

Для выполнения данной курсовой работы я предварительно реализовал 7 файлов с кодом:  
  
CMakeLists.txt - описание процесса сборки проекта  
  
mappedFile.h - реализация mapped file. Содержит структуру, в которой хранится файловый дескриптор и массив чаров.  
  
game.hpp - отдельный файл классов игрока и игры.  
  
mutex.h - заголовочный файл для общего мьютекса.  
  
mutex.cpp - реализация общего мьютекса для процессов.

server.cpp - реализация программы сервера.  
  
client.cpp - реализация программы клиента.

**Общий метод и алгоритм решения**

CMakeLists.txt

add\_executable(serverGame server.cpp include/game.hpp include/mutex.h include/mappedFile.h src/mutex.cpp)

add\_executable(clientGame client.cpp include/game.hpp include/mutex.h include/mappedFile.h src/mutex.cpp)

target\_include\_directories(serverGame PRIVATE include)

target\_include\_directories(clientGame PRIVATE include)

target\_link\_libraries(serverGame PRIVATE Threads::Threads)

target\_link\_libraries(clientGame PRIVATE Threads::Threads)

По сути, две работающие программы. В начале запускается сервер, после два клиента. При команде create создается игра. При команде connect игрок присоединяется к текущей игре. Далее при помощи внутриигровых команд shoot и stats игроки могут стрелять по чужому полю и смотреть свою статистику. Все действия обрабатываются на сервере.

**Исходный код  
mappedFile.h  
#ifndef MAPPED\_FILE\_H**

**#define MAPPED\_FILE\_H**

**#define \_MAPPED\_SIZE 8192**

**#define \_SHM\_OPEN\_MODE S\_IWUSR | S\_IRUSR | S\_IRGRP | S\_IROTH**

**#define \_BUFFER\_NAME "mybuffer.buf"**

**#define \_MUTEX\_NAME "mymutex.mutex"**

**#define \_MSG\_SEP '$'**

**struct TMappedFile {**

**int fd;**

**char \*data;**

**};**

**#endif**

**game.hpp  
#ifndef PLAYERANDGAME\_H**

**#define PLAYERANDGAME\_H**

**#include <algorithm>**

**#include <vector>**

**#include <string>**

**#include <iostream>**

**class TPlayer {**

**public:**

**std::string username;**

**std::vector<std::vector<char>> field;**

**int wins{};**

**int loses{};**

**int kills{};**

**int misses{};**

**int wounds{};**

**bool turn{};**

**TPlayer() : field(12, std::vector<char> (12, '.')) {}**

**void ErasePlayer() {**

**username = "";**

**wins = 0;**

**loses = 0;**

**kills = 0;**

**misses = 0;**

**wounds = 0;**

**turn = false;**

**}**

**};**

**class TGame {**

**public:**

**std:: string name;**

**std:: string password;**

**bool connected{};**

**bool created{};**

**void EraseGame() {**

**name = "";**

**password = "";**

**connected = false;**

**created = false;**

**}**

**};**

**void RandomLocation(std::vector<std::vector<char>> &field) {**

**int j =- 1;**

**int k;**

**int v;**

**int l;**

**int x[2];**

**int 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) {**

**std::replace(field[i].begin(), field[i].end(), '/', '.');**

**}**

**}**

**void PrintField(std::vector<std::vector<char>> &field) {**

**for (int i = 1; i < 11; ++i) {**

**for (int j = 1; j < 11; ++j) {**

**std:: cout << field[i][j];**

**}**

**std:: cout << std:: endl;**

**}**

**}**

**bool WonGame(std::vector<std::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(std::vector<std::vector<char>>& field) {**

**for (int i = 0; i < 12; i++) {**

**field[i].clear();**

**field[i] = std::vector<char>(12, '.');**

**}**

**}**

**#endif**

**mutex.h**

**#ifndef SHARED\_MUTEX\_H**

**#define SHARED\_MUTEX\_H**

**#include <pthread.h>**

**struct TCommonMutex {**

**pthread\_mutex\_t \*ptr; // Pointer to the pthread mutex and shared memory segment**

**int shm\_fd; // Descriptor of shared memory object**

**char \*name; // Name of the mutex and associated shared memory object**

**int created; // 1 if created new mutex, 0 if mutex was retrieved from memory**

**};**

**TCommonMutex SharedMutexInit(const char \*name);**

**int SharedMutexDestroy(TCommonMutex mutex);**

**#endif**

**mutex.cpp  
#include <cerrno>**

**#include <fcntl.h>**

**#include <linux/limits.h>**

**#include <cstdio>**

**#include <cstdlib>**

**#include <cstring>**

**#include <sys/mman.h>**

**#include <unistd.h>**

**#include <iostream>**

**#include "mutex.h"**

**TCommonMutex SharedMutexInit(const char \*name) {**

**TCommonMutex mutex = {nullptr, 0, nullptr, 0};**

**errno = 0;**

**mutex.shm\_fd = shm\_open(name, O\_RDWR, 0660);**

**if (errno == ENOENT) {**

**mutex.shm\_fd = shm\_open(name, O\_RDWR | O\_CREAT, 0660);**

**mutex.created = 1;**

**}**

**if (mutex.shm\_fd == -1) {**

**std:: cout << "An error while shm\_open has been detected!" << std:: endl;**

**return mutex;**

**}**

**if (ftruncate(mutex.shm\_fd, sizeof(pthread\_mutex\_t)) != 0) {**

**std:: cout << "An error while ftruncate has been detected!" << std:: endl;**

**return mutex;**

**}**

**void \*address = mmap(nullptr, sizeof(pthread\_mutex\_t), PROT\_READ | PROT\_WRITE, MAP\_SHARED, mutex.shm\_fd, 0);**

**if (address == MAP\_FAILED) {**

**std:: cout << "An error with mmaping has been detected!" << std:: endl;**

**return mutex;**

**}**

**auto \*mutexPtr = (pthread\_mutex\_t \*)address;**

**// If shared memory was just created -- initialize the mutex as well.**

**if (mutex.created != 0) {**

**pthread\_mutexattr\_t attr; // Deadlock to common shared data!**

**if (pthread\_mutexattr\_init(&attr) != 0) {**

**std:: cout << "An error while pthread\_mutexattr\_init has been detected!" << std:: endl;**

**return mutex;**

**}**

**if (pthread\_mutexattr\_setpshared(&attr, PTHREAD\_PROCESS\_SHARED) != 0) { // PTHREAD\_PROCESS\_SHARED - may be operated on by any thread in any process that has access to it**

**std:: cout << "An error while pthread\_mutexattr\_setpshared has been detected!" << std:: endl;**

**return mutex;**

**} //pthread\_mutexattr\_setpsharedshall set the process-shared attribute in an initialized attributes object referenced by attr.**

**if (pthread\_mutex\_init(mutexPtr, &attr) != 0) {**

**std:: cout << "An error while pthread\_mutex\_init has been detected!" << std:: endl;**

**return mutex;**

**}**

**}**

**mutex.ptr = mutexPtr;**

**mutex.name = (char \*)malloc(NAME\_MAX + 1);**

**strcpy(mutex.name, name);**

**return mutex;**

**}**

**int SharedMutexDestroy(TCommonMutex mutex) {**

**if ((errno = pthread\_mutex\_destroy(mutex.ptr)) != 0) {**

**std:: cout << "An error while destroying mutex has been detected!" << std:: endl;**

**return -1;**

**}**

**if (munmap((void \*)mutex.ptr, sizeof(pthread\_mutex\_t)) != 0) {**

**std:: cout << "An error while munmap has been detected!" << std:: endl;**

**return -1;**

**}**

**mutex.ptr = nullptr;**

**if (close(mutex.shm\_fd) != 0) {**

**std:: cout << "An error while closing has been detected!" << std:: endl;**

**return -1;**

**}**

**mutex.shm\_fd = 0;**

**if (shm\_unlink(mutex.name) != 0) {**

**std:: cout << "An error while shm\_unlink has been detected!" << std:: endl;**

**return -1;**

**}**

**free(mutex.name);**

**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 <algorithm>**

**#include <sys/stat.h>**

**#include <fstream>**

**#include "mappedFile.h"**

**#include "game.hpp"**

**#include "mutex.h"**

**TMappedFile mappedFile;**

**TCommonMutex mutex;**

**std:: string nickname;**

**bool playing = false;**

**std:: string currentGame;**

**void SendMessage (const std:: string &message) {**

**if (pthread\_mutex\_lock(mutex.ptr) != 0) {**

**std:: cout << "An error while locking mutex has been detected!" << std:: endl;**

**exit(EXIT\_FAILURE);**

**}**

**memset(mappedFile.data, '\0', \_MAPPED\_SIZE);**

**sprintf(mappedFile.data, "%s", message.c\_str());**

**pthread\_mutex\_unlock(mutex.ptr);**

**}**

**bool ReceiveAnswer() {**

**if (mappedFile.data[0] != 'T' || mappedFile.data[1] != 'O' || mappedFile.data[2] != \_MSG\_SEP) {**

**return false;**

**}**

**std:: string message = mappedFile.data;**

**std:: vector<std:: string> serverCommands;**

**std:: string string;**

**for (char i : message) {**

**if (i == \_MSG\_SEP) {**

**serverCommands.push\_back(string);**

**string = "";**

**}**

**else {**

**string.push\_back(i);**

**}**

**}**

**if (serverCommands[1] == nickname) {**

**if (pthread\_mutex\_lock(mutex.ptr) != 0) {**

**std:: cout << "An error while locking mutex has been detected!" << std:: endl;**

**exit(EXIT\_FAILURE);**

**}**

**memset(mappedFile.data, '\0', \_MAPPED\_SIZE);**

**pthread\_mutex\_unlock(mutex.ptr);**

**if (serverCommands[2] == "gamecreated") {**

**playing = true;**

**std:: cout << "Created successfully!" << std:: endl;**

**std:: cout << "You are a player №1, cause you have created the game. Your field has been prepared!" << std:: endl;**

**return true;**

**}**

**if (serverCommands[2] == "connected") {**

**std:: cout << "Connected sucessfully" << std:: endl;**

**std:: cout << "You are a player №2, cause you have connected to the game. Your field has been prepared!" << std:: endl;**

**playing = true;**

**return true;**

**}**

**if (serverCommands[2] == "notatgame") {**

**playing = true;**

**std:: cout << "You can't play without another player!" << std:: endl;;**

**return true;**

**}**

**if (serverCommands[2] == "gamenotexists") {**

**std:: cout << "Game with this name not exists" << std:: endl;**

**playing = false;**

**currentGame = "";**

**return true;**

**}**

**if (serverCommands[2] == "wrongpassword") {**

**std:: cout << "Wrong password has been detected!" << std:: endl;**

**playing = false;**

**currentGame = "";**

**return true;**

**}**

**if (serverCommands[2] == "notyourturn") {**

**std:: cout << "It's not your turn now!" << std:: endl;**

**playing = true;**

**return true;**

**}**

**if (serverCommands[2] == "youwounded") {**

**playing = true;**

**std:: cout << "You have wounded enemy's ship! Please enter coordinates again!" << std:: endl;**

**return true;**

**}**

**if (serverCommands[2] == "youmissed") {**

**playing = true;**

**std:: cout << "Unfortunately you have missed! Now it's your enemy's turn!" << std:: endl;**

**return true;**

**}**

**if (serverCommands[2] == "youkilled") {**

**playing = true;**

**std:: cout << "Congrats, you have KILLED enemy's ship! Please enter coordinates again!" << std:: endl;**

**return true;**

**}**

**if (serverCommands[2] == "zeroplaces") {**

**playing = false;**

**std:: cout << "Sorry, but you can not create a game or connect to existing game. There are not free places!" << std:: endl;**

**return true;**

**}**

**if (serverCommands[2] == "yourepeated") {**

**playing = true;**

**std:: cout << "You have already entered these coordinates! Please enter something new." << std:: endl;**

**return true;**

**}**

**if (serverCommands[2] == "disconnected") {**

**std:: cout << "You have successfully disconnected from the server!" << std:: endl;**

**playing = false;**

**return true;**

**}**

**if (serverCommands[2] == "youwon") {**

**std:: cout << "YOU WON THE GAME!" << std:: endl;**

**playing = false;**

**return true;**

**}**

**if (serverCommands[2] == "stats") {**

**int wins = stoi(serverCommands[3]);**

**int loses = stoi(serverCommands[4]);**

**int kills = stoi(serverCommands[5]);**

**int misses = stoi(serverCommands[6]);**

**int wounds = stoi(serverCommands[7]);**

**std:: cout << "You have " << wins << " wins and " << loses << " loses!" << std:: endl;**

**std:: cout << "FULL STATISTICS: " << std:: endl;**

**std:: cout << '\t' << kills << " kills" << std:: endl;**

**std:: cout << '\t' << wounds << " wounds" << std:: endl;**

**std:: cout << '\t' << misses << " misses" << std:: endl;**

**playing = true;**

**return true;**

**}**

**else {**

**std:: cout << "Warning: unknown message has been detected!" << std::endl;**

**playing = false;**

**return true;**

**}**

**return true;**

**}**

**return false;**

**}**

**void Help() {**

**std:: cout << "Follow next rules: " << std:: endl;**

**std:: cout << '\t' << "create for creating a new game" << std:: endl;**

**std:: cout << '\t' << "connect for connecting to the server" << std:: endl;**

**std:: cout << '\t' << "shoot for shooting at enemy's ship" << std:: endl;**

**std:: cout << '\t' << "stats for checking your stats" << std:: endl;**

**std:: cout << '\t' << "disconnect for leaving from the server" << std:: endl;**

**std:: cout << '\t' << "quit for leaving from the program" << std:: endl;**

**std:: cout << '\t' << "help for checking rules" << std:: endl;**

**}**

**int main() {**

**mappedFile.fd = shm\_open(\_BUFFER\_NAME, O\_RDWR, \_SHM\_OPEN\_MODE);**

**if (mappedFile.fd == -1 ) {**

**std:: cout << "An error while shm\_open has been detected!" << std:: endl;**

**exit(EXIT\_FAILURE);**

**}**

**mutex = SharedMutexInit(\_MUTEX\_NAME);**

**mappedFile.data = (char\*)mmap(0, \_MAPPED\_SIZE, PROT\_READ | PROT\_WRITE, MAP\_SHARED, mappedFile.fd, 0);**

**if (mappedFile.data == MAP\_FAILED) {**

**std:: cout << "An error while mmaping has been detected!" << std:: endl;**

**}**

**std:: cout << "Welcome to the SeaBattle! Please enter your nickname: " << std:: endl;**

**std:: cout << "> ";**

**std:: cin >> nickname;**

**std:: cout << "Hello, " << nickname << "!" << std::endl;**

**Help();**

**std:: string command;**

**while (std:: cout << "> " && std:: cin >> command) {**

**if (!playing && command == "create") {**

**std:: string gamename;**

**std:: string password;**

**std:: cin >> gamename >> password;**

**currentGame = gamename;**

**std::string on = "ON";**

**std:: string serverMessage = on + \_MSG\_SEP + nickname + \_MSG\_SEP + "create" + \_MSG\_SEP + gamename + \_MSG\_SEP + password + \_MSG\_SEP;**

**SendMessage (serverMessage);**

**bool hasnotanswer = true;**

**while (hasnotanswer) {**

**hasnotanswer = !ReceiveAnswer();**

**}**

**}**

**else if (playing && command == "create") {**

**std:: string gamename;**

**std:: string password;**

**std:: cin >> gamename >> password;**

**std:: cout << "Can't create a new game, you are playing now! Please enter another command!" << std:: endl;**

**continue;**

**}**

**else if (!playing && command == "connect") {**

**std:: string gamename;**

**std:: string password;**

**std:: cin >> gamename >> password;**

**currentGame = gamename;**

**std::string on = "ON";**

**std:: string serverMessage = on + \_MSG\_SEP + nickname + \_MSG\_SEP + "connect" + \_MSG\_SEP + gamename + \_MSG\_SEP + password + \_MSG\_SEP;**

**SendMessage (serverMessage);**

**bool hasnotanswer = true;**

**while (hasnotanswer) {**

**hasnotanswer = !ReceiveAnswer();**

**}**

**}**

**else if (playing && command == "connect") {**

**std:: string gamename;**

**std:: string password;**

**std:: cin >> gamename >> password;**

**std:: cout << "Can't connect to a new game, you've already connected! Please enter another command!" << std:: endl;**

**continue;**

**}**

**else if (playing && command == "shoot") {**

**int number;**

**char letter;**

**std:: cin >> letter >> number;**

**if ((!((letter >= 'A') && (letter <= 'J'))) || ((number < 1) || (number > 10))) {**

**std:: cout << "Please enter letter between A and J and number between 1 and 10!" << std:: endl;**

**continue;**

**}**

**else {**

**std:: string on = "ON";**

**std:: string serverMessage = on + \_MSG\_SEP + nickname + \_MSG\_SEP + "shoot" + \_MSG\_SEP + currentGame + \_MSG\_SEP + letter + \_MSG\_SEP + std:: to\_string(number) + \_MSG\_SEP;**

**SendMessage (serverMessage);**

**bool hasnotanswer = true;**

**while (hasnotanswer) {**

**hasnotanswer = !ReceiveAnswer();**

**}**

**}**

**}**

**else if (playing && command == "stats") {**

**std::string on = "ON";**

**std:: string serverMessage = on + \_MSG\_SEP + nickname + \_MSG\_SEP + "stats" + \_MSG\_SEP + currentGame + \_MSG\_SEP;**

**SendMessage (serverMessage);**

**bool hasnotanswer = true;**

**while (hasnotanswer) {**

**hasnotanswer = !ReceiveAnswer();**

**}**

**}**

**else if (!playing && command == "shoot") {**

**int number;**

**char letter;**

**std:: cin >> letter >> number;**

**std:: cout << "You are not in the game right now. Please create a game or connect to the existing one!" << std:: endl;**

**continue;**

**}**

**else if (playing && command == "disconnect") {**

**std:: string on = "ON";**

**std:: string serverMessage = on + \_MSG\_SEP + nickname + \_MSG\_SEP + "disconnect" + \_MSG\_SEP + currentGame + \_MSG\_SEP;**

**SendMessage (serverMessage);**

**bool hasnotanswer = true;**

**while (hasnotanswer) {**

**hasnotanswer = !ReceiveAnswer();**

**}**

**}**

**else if (command == "help") {**

**Help();**

**}**

**else if (!playing && command == "quit") {**

**break;**

**}**

**else {**

**std:: cout << "Wrong input!" << std:: endl;**

**}**

**}**

**return 0;**

**}**

**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 <fstream>

#include "game.hpp"

#include "mappedFile.h"

#include "mutex.h"

int main() {

TPlayer creator;

TPlayer connector;

TGame game;

TMappedFile mappedFile;

std:: string clientMessage;

mappedFile.fd = shm\_open(\_BUFFER\_NAME, O\_RDWR | O\_CREAT, \_SHM\_OPEN\_MODE);

if (mappedFile.fd == -1) {

std:: cout << "Error with shm\_open function has been detected!" << std:: endl;

exit(EXIT\_FAILURE);

}

if (ftruncate(mappedFile.fd, \_MAPPED\_SIZE) == -1) {

std:: cout << "An error while ftruncate has been detected!" << std:: endl;

exit(EXIT\_FAILURE);

}

mappedFile.data = (char \*)mmap(0, \_MAPPED\_SIZE, PROT\_READ | PROT\_WRITE, MAP\_SHARED, mappedFile.fd, 0);

if (mappedFile.data == MAP\_FAILED) {

std:: cout << "An error with mmap function has been detected!" << std:: endl;

exit(EXIT\_FAILURE);

}

memset(mappedFile.data, '\0', \_MAPPED\_SIZE);

TCommonMutex mutex = SharedMutexInit(\_MUTEX\_NAME);

if (mutex.created == 0) {

std:: cout << "FROM SERVER: Mutex has been already created!" << std:: endl;

}

else {

errno = 0;

}

std:: cout << "Server is working now! Please start a game and it will be displayed here!" << std:: endl;

while (true) {

if (mappedFile.data[0] == EOF) {

break;

}

if (mappedFile.data[0] == '\0') {

continue;

}

if (!(mappedFile.data[0] == 'O' && mappedFile.data[1] == 'N' &&

mappedFile.data[2] == \_MSG\_SEP)) {

continue;

}

std:: cout << "FROM SERVER: Locking mutex" << std:: endl;

if (pthread\_mutex\_lock(mutex.ptr) != 0) {

std:: cout << "An error while locking mutex has been detected!" << std:: endl;

exit(EXIT\_FAILURE);

}

clientMessage = mappedFile.data;

std:: cout << "FROM SERVER: Has received next message from client: " << clientMessage << std:: endl;

memset(mappedFile.data, '\0', \_MAPPED\_SIZE);

std:: vector<std:: string> clientCommands;

std:: string string;

for (char i : clientMessage) {

if (i == \_MSG\_SEP) {

clientCommands.push\_back(string);

string = "";

}

else {

string.push\_back(i);

}

}

if (clientCommands[2] == "create") {

if (game.created || game.name == clientCommands[3]) {

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "zeroplaces" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

else {

game.created = true;

creator.turn = true;

connector.turn = false;

creator.username = clientCommands[1];

RandomLocation(creator.field);

game.name = clientCommands[3];

game.password = clientCommands[4];

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "gamecreated" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

}

else if (clientCommands[2] == "connect") {

if (game.connected) {

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "zeroplaces" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

else {

if (game.name == clientCommands[3]) {

if (game.password == clientCommands[4]) {

game.connected = true;

connector.turn = false;

creator.turn = true;

connector.username = clientCommands[1];

RandomLocation(connector.field);

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "connected" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

else {

game.connected = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "wrongpassword" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

}

else {

game.connected = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "gamenotexists" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message:" << playerMessage << std:: endl;

}

}

}

else if (clientCommands[2] == "shoot") {

if (!game.connected) {

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "notatgame" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

if (clientCommands[1] == connector.username) {

if (connector.turn && !creator.turn) {

if (game.name == clientCommands[3]) {

int number = std:: stoi(clientCommands[5]);

std:: string l = clientCommands[4];

char letter = l[0];

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.wounds++;

connector.kills++;

connector.turn = true;

creator.turn = false;

if (WonGame(creator.field)) {

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "youwon" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to connector next message:" << playerMessage << std:: endl;

connector.wins++;

creator.loses++;

std:: ofstream fout("Statistics.txt", std::ios\_base::app);

fout << connector.username << ": " << connector.wins << " wins, " << connector.loses << " loses, " << connector.kills << " kills, " << connector.misses << " misses, " << connector.wounds << " wounds, " << std:: endl;

fout << creator.username << ": " << creator.wins << " wins, " << creator.loses << " loses, " << creator.kills << " kills, " << creator.misses << " misses, " << creator.wounds << " wounds, " << std:: endl;

creator.ErasePlayer();

connector.ErasePlayer();

PrepareField(creator.field);

PrepareField(connector.field);

game.EraseGame();

}

else {

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "youkilled" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message:" << playerMessage << std:: endl;

}

}

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;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "yourepeated" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message:" << playerMessage << std:: endl;

}

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.wounds++;

connector.turn = true;

creator.turn = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "youwounded" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

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.wounds++;

connector.turn = true;

creator.turn = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "youwounded" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

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.wounds++;

connector.turn = true;

creator.turn = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "youwounded" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

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.wounds++;

connector.turn = true;

creator.turn = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] +

\_MSG\_SEP + "youwounded" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

else if (creator.field[number][int(letter) - int('A') + 1] == '.') {

connector.misses++;

connector.turn = false;

creator.turn = true;

creator.field[number][int(letter) - int('A') + 1] = 'm';

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "youmissed" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

std:: cout << "Current state of " << creator.username << "'s field is: " << std:: endl;

PrintField(creator.field);

}

else {

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "gamenotexists" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

}

else {

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "notyourturn" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

}

else if (clientCommands[1] == creator.username) {

if (creator.turn && !connector.turn) {

if (game.name == clientCommands[3]) {

int number = std::stoi(clientCommands[5]);

std:: string l = clientCommands[4];

char letter = l[0];

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.kills++;

creator.wounds++;

creator.turn = true;

connector.turn = false;

if (WonGame(connector.field)) {

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "youwon" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to creator next message: " << playerMessage << std:: endl;

creator.wins++;

connector.loses++;

std:: ofstream fout("Statistics.txt", std::ios\_base::app);

fout << connector.username << ": " << connector.wins << " wins, " << connector.loses << " loses, " << connector.kills << " kills, " << connector.misses << " misses, " << connector.wounds << " wounds." << std:: endl;

fout << creator.username << ": " << creator.wins << " wins, " << creator.loses << " loses, " << creator.kills << " kills, " << creator.misses << " misses, " << creator.wounds << " wounds. " << std:: endl;

creator.ErasePlayer();

connector.ErasePlayer();

PrepareField(creator.field);

PrepareField(connector.field);

game.EraseGame();

}

else {

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "youkilled" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

}

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;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "yourepeated" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

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.wounds++;

creator.turn = true;

connector.turn = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "youwounded" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

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.wounds++;

creator.turn = true;

connector.turn = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "youwounded" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

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.wounds++;

creator.turn = true;

connector.turn = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "youwounded" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

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.wounds++;

connector.turn = true;

creator.turn = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] +

\_MSG\_SEP + "youwounded" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

else if (connector.field[number][int(letter) - int('A') + 1] == '.') {

creator.misses++;

creator.turn = false;

connector.turn = true;

connector.field[number][int(letter) - int('A') + 1] = 'm';

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "youmissed" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

std:: cout << "Current state of " << connector.username << "'s field is: " << std:: endl;

PrintField(connector.field);

}

else {

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "gamenotexists" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

}

else {

creator.turn = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "notyourturn" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

}

}

else if (clientCommands[2] == "disconnect") {

if (clientCommands[1] == creator.username) {

creator.turn = false;

connector.turn = true;

game.connected = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "disconnected" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std::cout << "FROM SERVER: Sending to client next message: " << playerMessage << std::endl;

game.created = false;

}

else {

creator.turn = true;

connector.turn = false;

game.connected = false;

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + connector.username + \_MSG\_SEP + "disconnected" + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std::cout << "FROM SERVER: Sending to client next message: " << playerMessage << std:: endl;

}

}

else if (clientCommands[2] == "stats") {

if (creator.username == clientCommands[1]) {

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "stats" + \_MSG\_SEP + std:: to\_string(creator.wins) + \_MSG\_SEP + std:: to\_string(creator.loses) + \_MSG\_SEP + std:: to\_string(creator.kills) + \_MSG\_SEP + std:: to\_string(creator.misses) + \_MSG\_SEP + std:: to\_string(creator.wounds) + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to creator next message: " << playerMessage << std:: endl;

}

else {

std:: string to = "TO";

std:: string playerMessage = to + \_MSG\_SEP + clientCommands[1] + \_MSG\_SEP + "stats" + \_MSG\_SEP + std:: to\_string(connector.wins) + \_MSG\_SEP + std:: to\_string(connector.loses) + \_MSG\_SEP + std:: to\_string(connector.kills) + \_MSG\_SEP + std:: to\_string(connector.misses) + \_MSG\_SEP + std:: to\_string(connector.wounds) + \_MSG\_SEP;

sprintf(mappedFile.data, "%s", playerMessage.c\_str());

std:: cout << "FROM SERVER: Sending to connector next message: " << playerMessage << std::endl;

}

}

pthread\_mutex\_unlock(mutex.ptr);

std:: cout << "FROM SERVER: Unlocked mutex" << std:: endl;

}

if (SharedMutexDestroy(mutex) == -1) {

std:: cout << "An error while destroying mutex has been detected!" << std:: endl;

exit(EXIT\_FAILURE);

}

if (shm\_unlink(\_BUFFER\_NAME) == -1) {

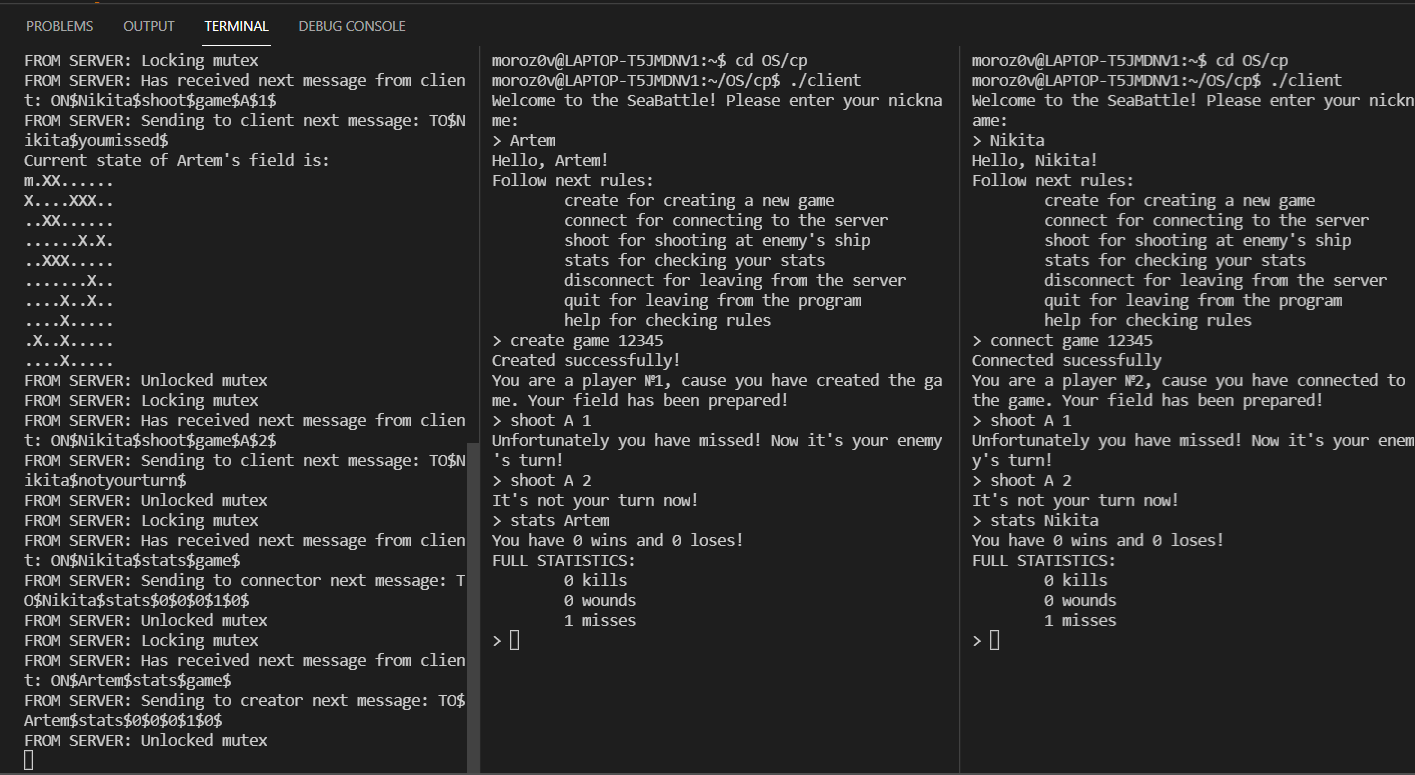
std:: cout << "An error while shm\_unlink has been detected!" << std:: endl;

exit(EXIT\_FAILURE);

}

return 0;

}

**Демонстрация работы программы  
  
**

**Выводы**Курсовой проект, на мой взгляд, является отличным завершением курса “Операционные системы”. Благодаря нему я укрепил свои знания в этой сфере, поработав с примитивами синхронизации и мемори-маппингом.