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: Морозов Артем Борисович
Группа: М8О-208Б-20
Вариант: 5
Преподаватель: Миронов Евгений Сергеевич
Оценка: ___________
Дата: ___________
Подпись: ___________

Москва, 2021

Содержание

Репозиторий

https://github.com/artemmoroz0v

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 лабораторной работы я предварительно реализовал 7 файлов с кодом:

MappedFile.h - реализация mapped file. Содержит структуру, в которой хранится файловый дескриптор и массив чаров.

ZMQFunctions.h - отдельный файл для функций zero-message queue, сделанный для удобства работы и во избежание загрязнения кода.

CommonMutex.h - заголовочный файл для общего мьютекса.

CommonMutex.cpp - реализация общего мьютекса для процессов.

ServerProgram.cpp - реализация программы сервера.

ClientProgram.cpp - реализация программы клиента.

Общий метод и алгоритм решения

В makefile у нас две команды:

```
g++ ClientProgram.cpp CommonMutex.cpp -o client -lrt -pthread
g++ ServerProgram.cpp CommonMutex.cpp -o server -lrt -pthread
```

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 MappedFile {
    int fd;
    char *data;
};
#endif
```

PlayerAndGame.h

```
#ifndef PLAYERANDGAME_H
#define PLAYERANDGAME_H
#include <algorithm>
#include <vector>
class Player {
    public:
        std:: string username;
        std:: vector<std::vector<char>> field;
        int wins;
        int loses;
        int kills;
        int misses;
        int wounds;
        bool turn;
        Player() : wins(0), loses(0), kills(0), misses(0), wounds(0), field(12, std::vector<char> (12, '.')),
username(""), turn(false) {}
        void ErasePlayer() {
            username = "";
            wins = 0;
            loses = 0;
            kills = 0;
            misses = 0;
            wounds = 0;
            turn = false;
        }
};
class Game {
    public:
        std:: string name;
        std:: string password;
        bool connected;
        bool created;
        Game() : name(""), password(""), connected(false), created(false) {}
        void EraseGame() {
            name = "";
            password = "";
            connected = false;
            created = false;
        }
};
void RandomLocation (std::vector<std::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) {
        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
```

CommonMutex.h

```
#ifndef SHARED_MUTEX_H
#define SHARED_MUTEX_H
#include <pthread.h>
struct CommonMutex {
   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
};
// If mutex with name exists it will be loaded, otherwise mutex will be created
CommonMutex shared_mutex_init(const char *name);
// Close and destroy shared mutex and returns 0 in case of success, otherwise returns -1
int shared_mutex_destroy(CommonMutex mutex);
#endif
```

CommonMutex.cpp

```
#include "CommonMutex.h"
#include <errno.h>
#include <fcntl.h>
#include <linux/limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <unistd.h>
#include <iostream>

CommonMutex shared_mutex_init(const char *name) {
   CommonMutex mutex = {NULL, 0, NULL, 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(NULL, 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;
    }
    pthread_mutex_t *mutex_ptr = (pthread_mutex_t *)address;
    // If shared memory was just created -- initialize the mutex as well.
    if (mutex.created) {
        pthread_mutexattr_t attr; // Deadlock to common shared data!
        if (pthread_mutexattr_init(&attr)) {
            std:: cout << "An error while pthread_mutexattr_init has been detected!" << std:: endl;
            return mutex;
        }
        if (pthread_mutexattr_setpshared(&attr, PTHREAD_PROCESS_SHARED)) { //
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(mutex_ptr, &attr)) {
            std:: cout << "An error while pthread_mutex_init has been detected!" << std:: endl;
            return mutex;
        }
    }
    mutex.ptr = mutex_ptr;
    mutex.name = (char *)malloc(NAME_MAX + 1);
    strcpy(mutex.name, name);
    return mutex;
}
int shared_mutex_destroy(CommonMutex mutex) {
    if ((errno = pthread_mutex_destroy(mutex.ptr))) {
        std:: cout << "An error while destroying mutex has been detected!" << std:: endl;
        return -1;
    }
    if (munmap((void *)mutex.ptr, sizeof(pthread_mutex_t))) {
        std:: cout << "An error while munmap has been detected!" << std:: endl;
        return -1;
    }
    mutex.ptr = NULL;
    if (close(mutex.shm_fd)) {
        std:: cout << "An error while closing has been detected!" << std:: endl;
        return -1;
    }
    mutex.shm_fd = 0;
    if (shm_unlink(mutex.name)) {
        std:: cout << "An error while shm_unlink has been detected!" << std:: endl;
        return -1;
    }
    free(mutex.name);
    return 0;
}
```

ClientProgram.cpp

```
#include <iostream>
#include <fcntl.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/mman.h>
#include <cassert>
#include <cstring>
#include <vector>
#include "MappedFile.h"
#include "CommonMutex.h"
#include <algorithm>
```

```
#include <sys/stat.h>
#include <fstream>
MappedFile mapped_file;
CommonMutex mutex;
std:: string nickname;
bool playing = false;
std:: string current_game = "";

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(mapped_file.data, '\0', _MAPPED_SIZE);
    sprintf(mapped_file.data, "%s", message.c_str());
    pthread_mutex_unlock(mutex.ptr);
}

bool ReceiveAnswer() {
    if (mapped_file.data[0] != 'T' || mapped_file.data[1] != 'O' || mapped_file.data[2] != _MSG_SEP) {
        return false;
    }
    std:: string message = mapped_file.data;
    std:: vector<std:: string> server_commands;
    std:: string string = "";
    for (int i = 0; i < message.size(); i++) {
        if (message[i] == _MSG_SEP) {
            server_commands.push_back(string);
            string = "";
        }
        else {
            string.push_back(message[i]);
        }
    }
    if (server_commands[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(mapped_file.data, '\0', _MAPPED_SIZE);
        pthread_mutex_unlock(mutex.ptr);
        if (server_commands[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 (server_commands[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 (server_commands[2] == "notatgame") {
            playing = true;
            std:: cout << "You can't play without another player!" << std:: endl;;
            return true;
        }
        if (server_commands[2] == "gamenotexists") {
            std:: cout << "Game with this name not exists" << std:: endl;
            playing = false;
            current_game = "";
            return true;
        }
        if (server_commands[2] == "wrongpassword") {
            std:: cout << "Wrong password has been detected!" << std:: endl;
            playing = false;
            current_game = "";
            return true;
        }
        if (server_commands[2] == "notyourturn") {
            std:: cout << "It's not your turn now!" << std:: endl;
            playing = true;
            return true;
        }
        if (server_commands[2] == "youwounded") {
            playing = true;
            std:: cout << "You have wounded enemy's ship! Please enter coordinates again!" << std:: endl;
            return true;
        }
        if (server_commands[2] == "youmissed") {
            playing = true;
            std:: cout << "Unfortunately you have missed! Now it's your enemy's turn!" << std:: endl;
            return true;
        }
        if (server_commands[2] == "youkilled") {
            playing = true;
            std:: cout << "Congrats, you have KILLED enemy's ship! Please enter coordinates again!" << std::
endl;
            return true;
        }
        if (server_commands[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 (server_commands[2] == "yourepeated") {
            playing = true;
            std:: cout << "You have already entered these coordinates! Please enter something new." << std::
endl;
            return true;
        }
        if (server_commands[2] == "disconnected") {
            std:: cout << "You have successfully disconnected from the server!" << std:: endl;
            playing = false;
            return true;
        }
        if (server_commands[2] == "youwon") {
            std:: cout << "YOU WON THE GAME!" << std:: endl;
            playing = false;
            return true;
        }
        if (server_commands[2] == "stats") {
            int wins = stoi(server_commands[3]);
            int loses = stoi(server_commands[4]);
            int kills = stoi(server_commands[5]);
            int misses = stoi(server_commands[6]);
            int wounds = stoi(server_commands[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() {
    mapped_file.fd = shm_open(_BUFFER_NAME, O_RDWR, _SHM_OPEN_MODE);
    if (mapped_file.fd == -1 ) {
        std:: cout << "An error while shm_open has been detected!" << std:: endl;
        exit(EXIT_FAILURE);
    }
    mutex = shared_mutex_init(_MUTEX_NAME);
    mapped_file.data = (char*)mmap(0, _MAPPED_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, mapped_file.fd, 0);
    if (mapped_file.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, password;
            std:: cin >> gamename >> password;
            current_game = gamename;
            std::string on = "ON";
            std:: 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") {
            std:: string gamename, 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, password;
            std:: cin >> gamename >> password;
            current_game = gamename;
            std::string on = "ON";
            std:: 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") {
            std:: string gamename, 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 server_message = on + _MSG_SEP + nickname + _MSG_SEP + "shoot" +
_MSG_SEP + current_game + _MSG_SEP + letter + _MSG_SEP + std:: to_string(number) +
_MSG_SEP;
```

```
            SendMessage (server_message);
            bool hasnotanswer = true;
            while (hasnotanswer) {
                hasnotanswer = !ReceiveAnswer();
            }
        }
    }
    else if (playing && command == "stats") {
        std:: string username;
        std:: cin >> username;
        std::string on = "ON";
        std:: string server_message = on + _MSG_SEP + username + _MSG_SEP + "stats" + _MSG_SEP
+ current_game + _MSG_SEP;
        SendMessage (server_message);
        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 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 {
            std:: cout << "Wrong input!" << std:: endl;
        }
    }
    return 0;
}
```

ServerProgram.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.h"
#include "CommonMutex.h"
#include "PlayerAndGame.h"
#include <fstream>

int main() {
    Player creator;
    Player connector;
    Game game;
    MappedFile mapped_file;
    std:: string client_message = "";
    mapped_file.fd = shm_open(_BUFFER_NAME, O_RDWR | O_CREAT, _SHM_OPEN_MODE);
    if (mapped_file.fd == -1) {
        std:: cout << "Error with shm_open function has been detected!" << std:: endl;
        exit(EXIT_FAILURE);
    }
    if (ftruncate(mapped_file.fd, _MAPPED_SIZE) == -1) {
        std:: cout << "An error while ftruncate has been detected!" << std:: endl;
        exit(EXIT_FAILURE);
    }
    mapped_file.data = (char *)mmap(0, _MAPPED_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, mapped_file.fd, 0);
    if (mapped_file.data == MAP_FAILED) {
        std:: cout << "An error with mmap function has been detected!" << std:: endl;
        exit(EXIT_FAILURE);
    }
    memset(mapped_file.data, '\0', _MAPPED_SIZE);
    CommonMutex mutex = shared_mutex_init(_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 (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;
        }
        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);
        }
        client_message = mapped_file.data;
        std:: cout << "FROM SERVER: Has received next message from client: " << client_message <<
std:: endl;
        memset(mapped_file.data, '\0', _MAPPED_SIZE);
        std:: vector<std:: string> client_commands;
        std:: string string = "";
        for (int i = 0; i < client_message.size(); ++i) {
            if (client_message[i] == _MSG_SEP) {
              client_commands.push_back(string);
              string = "";
            }
            else {
              string.push_back(client_message[i]);
            }
        }
        if (client_commands[2] == "create") {
            if (game.created || game.name == client_commands[3]) {
              std:: string to = "TO";
              std:: string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"zeroplaces" + _MSG_SEP;
              sprintf(mapped_file.data, "%s", player_message.c_str());
              std:: cout << "FROM SERVER: Sending to client next message: " << player_message << std::
endl;
            }
            else {
              game.created = true;
              creator.turn = true;
              connector.turn = false;
              creator.username = client_commands[1];
              RandomLocation(creator.field);
              game.name = client_commands[3];
              game.password = client_commands[4];
              std:: string to = "TO";
              std:: string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"gamecreated" + _MSG_SEP;
              sprintf(mapped_file.data, "%s", player_message.c_str());
```

```
            std:: cout << "FROM SERVER: Sending to client next message: " << player_message << std::
endl;
        }
    }
    else if (client_commands[2] == "connect") {
        if (game.connected) {
            std:: string to = "TO";
            std:: string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"zeroplaces" + _MSG_SEP;
            sprintf(mapped_file.data, "%s", player_message.c_str());
            std:: cout << "FROM SERVER: Sending to client next message: " << player_message << std::
endl;
        }
        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];
                    RandomLocation(connector.field);
                    std:: string to = "TO";
                    std:: string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"connected" + _MSG_SEP;
                    sprintf(mapped_file.data, "%s", player_message.c_str());
                    std:: cout << "FROM SERVER: Sending to client next message: " << player_message <<
std:: endl;
                }
                else {
                    game.connected = false;
                    std:: string to = "TO";
                    std:: string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"wrongpassword" + _MSG_SEP;
                    sprintf(mapped_file.data, "%s", player_message.c_str());
                    std:: cout << "FROM SERVER: Sending to client next message: " << player_message <<
std:: endl;
                }
            }
            else {
                game.connected = false;
                std:: string to = "TO";
                std:: string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"gamenotexists" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                std:: cout << "FROM SERVER: Sending to client next message:" << player_message << std::
endl;
            }
        }
    }
    else if (client_commands[2] == "shoot") {
        if (!game.connected) {
            std:: string to = "TO";
            std:: string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"notatgame" + _MSG_SEP;
            sprintf(mapped_file.data, "%s", player_message.c_str());
            std:: cout << "FROM SERVER: Sending to client next message: " << player_message << std::
endl;
        }
        if (client_commands[1] == connector.username) {
            if (connector.turn && !creator.turn) {
                if (game.name == client_commands[3]) {
                    int number = std:: stoi(client_commands[5]);
                    std:: string l = client_commands[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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP 
+ "youwon" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                std:: cout << "FROM SERVER: Sending to connector next message:" << 
player_message << 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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP 
+ "youkilled" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                std:: cout << "FROM SERVER: Sending to client next message:" << player_message 
<< 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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"yourepeated" + _MSG_SEP;
                        sprintf(mapped_file.data, "%s", player_message.c_str());
                        std:: cout << "FROM SERVER: Sending to client next message:" << player_message
<< 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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"youwounded" + _MSG_SEP;
                        sprintf(mapped_file.data, "%s", player_message.c_str());
                        std:: cout << "FROM SERVER: Sending to client next message: " << player_message
<< 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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"youwounded" + _MSG_SEP;
              sprintf(mapped_file.data, "%s", player_message.c_str());
              std:: cout << "FROM SERVER: Sending to client next message: " << player_message
<< 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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"youwounded" + _MSG_SEP;
              sprintf(mapped_file.data, "%s", player_message.c_str());
              std:: cout << "FROM SERVER: Sending to client next message: " << player_message
<< std:: endl;
```

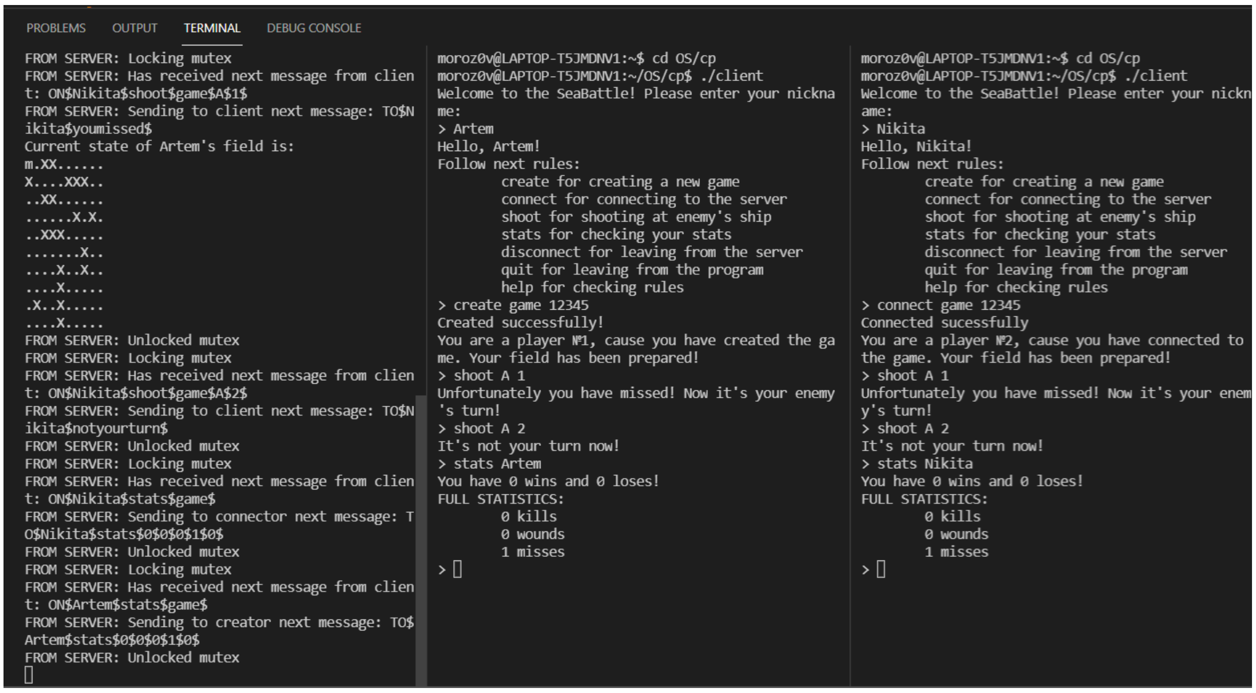
```
            }
            else if (creator.field[number][int(letter) - int('A') + 1] == 'X' && creator.field[number +
1][int(letter) - int('A') + 1] == 'X') {
                creator.field[number][int(letter) - int('A') + 1] = 'w';
                connector.wounds++;
                connector.turn = true;
                creator.turn = false;
                std:: string to = "TO";
                std:: string player_message = to + _MSG_SEP + client_commands[1] +
                              _MSG_SEP + "youwounded" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                std:: cout << "FROM SERVER: Sending to client next message: " << player_message
<< 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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"youmissed" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                std:: cout << "FROM SERVER: Sending to client next message: " << player_message
<< std:: endl;
            }
            std:: cout << "Current state of " << creator.username << "'s field is: " << std:: endl;
            PrintField(creator.field);
        }
        else {
            std:: string to = "TO";
            std:: string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"gamenotexists" + _MSG_SEP;
            sprintf(mapped_file.data, "%s", player_message.c_str());
            std:: cout << "FROM SERVER: Sending to client next message: " << player_message <<
std:: endl;
        }
```

```
            }
            else {
                std:: string to = "TO";
                std:: string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"notyourturn" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                std:: cout << "FROM SERVER: Sending to client next message: " << player_message <<
std:: endl;
            }
        }
        else if (client_commands[1] == creator.username) {
            if (creator.turn && !connector.turn) {
                if (game.name == client_commands[3]) {
                    int number = std::stoi(client_commands[5]);
                    std:: string l = client_commands[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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP
+ "youwon" + _MSG_SEP;
```

```
                        sprintf(mapped_file.data, "%s", player_message.c_str());
                        std:: cout << "FROM SERVER: Sending to creator next message: " <<
player_message << 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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP
+ "youkilled" + _MSG_SEP;
                        sprintf(mapped_file.data, "%s", player_message.c_str());
                        std:: cout << "FROM SERVER: Sending to client next message: " << player_message
<< 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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"yourepeated" + _MSG_SEP;
                     sprintf(mapped_file.data, "%s", player_message.c_str());
                     std:: cout << "FROM SERVER: Sending to client next message: " << player_message
<< 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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"youwounded" + _MSG_SEP;
                    sprintf(mapped_file.data, "%s", player_message.c_str());
                    std:: cout << "FROM SERVER: Sending to client next message: " << player_message
<< 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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"youwounded" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                std:: cout << "FROM SERVER: Sending to client next message: " << player_message
<< 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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"youwounded" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                std:: cout << "FROM SERVER: Sending to client next message: " << player_message
<< 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 player_message = to + _MSG_SEP + client_commands[1] +
                        _MSG_SEP + "youwounded" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                std:: cout << "FROM SERVER: Sending to client next message: " << player_message
<< 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 player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"youmissed" + _MSG_SEP;
                sprintf(mapped_file.data, "%s", player_message.c_str());
                std:: cout << "FROM SERVER: Sending to client next message: " << player_message
<< std:: endl;
            }
            std:: cout << "Current state of " << connector.username << "'s field is: " << std:: endl;
            PrintField(connector.field);
        }
        else {
            std:: string to = "TO";
            std:: string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"gamenotexists" + _MSG_SEP;
            sprintf(mapped_file.data, "%s", player_message.c_str());
            std:: cout << "FROM SERVER: Sending to client next message: " << player_message <<
std:: endl;
        }
    }
    else {
        creator.turn = false;
        std:: string to = "TO";
        std:: string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"notyourturn" + _MSG_SEP;
        sprintf(mapped_file.data, "%s", player_message.c_str());
        std:: cout << "FROM SERVER: Sending to client next message: " << player_message <<
std:: endl;
    }
    }
}
else if (client_commands[2] == "disconnect") {
    if (client_commands[1] == creator.username) {
        creator.turn = false;
        connector.turn = true;
        game.connected = false;
        std:: string to = "TO";
        std:: string player_message = to + _MSG_SEP + client_commands[1] + _MSG_SEP +
"disconnected" + _MSG_SEP;
        sprintf(mapped_file.data, "%s", player_message.c_str());
        std::cout << "FROM SERVER: Sending to client next message: " << player_message <<
std::endl;
    }
    else {
        creator.turn = true;
        connector.turn = false;
        game.connected = false;
        std:: string to = "TO";
        std:: string player_message = to + _MSG_SEP + connector.username + _MSG_SEP +
"disconnected" + _MSG_SEP;
        sprintf(mapped_file.data, "%s", player_message.c_str());
        std::cout << "FROM SERVER: Sending to client next message: " << player_message << std::
endl;
    }
}
else if (client_commands[2] == "stats") {
    if (creator.username == client_commands[1]) {
        std:: string to = "TO";
        std:: string player_message = to + _MSG_SEP + client_commands[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(mapped_file.data, "%s", player_message.c_str());
        std:: cout << "FROM SERVER: Sending to creator next message: " << player_message << std::
endl;
    }
    else {
        std:: string to = "TO";
        std:: string player_message = to + _MSG_SEP + client_commands[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(mapped_file.data, "%s", player_message.c_str());
        std:: cout << "FROM SERVER: Sending to connector next message: " << player_message <<
std::endl;
    }
}
pthread_mutex_unlock(mutex.ptr);
std:: cout << "FROM SERVER: Unlocked mutex" << std:: endl;
}
if (shared_mutex_destroy(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



```
FROM SERVER: Locking mutex
FROM SERVER: Has received next message from clien
t: ON$Nikita$shoot$game$A$1$
FROM SERVER: Sending to client next message: TO$N
ikita$youmissed$
Current state of Artem's field is:
m.XX......
X....XXX..
..XX......
......X.X.
..XXX.....
.......X..
....X..X..
....X.....
.X..X.....
....X.....
FROM SERVER: Unlocked mutex
FROM SERVER: Locking mutex
FROM SERVER: Has received next message from clien
t: ON$Nikita$shoot$game$A$2$
FROM SERVER: Sending to client next message: TO$N
ikita$notyourturn$
FROM SERVER: Unlocked mutex
FROM SERVER: Locking mutex
FROM SERVER: Has received next message from clien
t: ON$Nikita$stats$game$
FROM SERVER: Sending to connector next message: T
O$Nikita$stats$0$0$0$1$0$
FROM SERVER: Unlocked mutex
FROM SERVER: Locking mutex
FROM SERVER: Has received next message from clien
t: ON$Artem$stats$game$
FROM SERVER: Sending to creator next message: TO$
Artem$stats$0$0$0$1$0$
FROM SERVER: Unlocked mutex
```

```
moroz0v@LAPTOP-T5JMDNV1:~$ cd OS/cp
moroz0v@LAPTOP-T5JMDNV1:~/OS/cp$ ./client
Welcome to the SeaBattle! Please enter your nickna
me:
> Artem
Hello, Artem!
Follow next rules:
        create for creating a new game
        connect for connecting to the server
        shoot for shooting at enemy's ship
        stats for checking your stats
        disconnect for leaving from the server
        quit for leaving from the program
        help for checking rules
> create game 12345
Created successfully!
You are a player №1, cause you have created the ga
me. Your field has been prepared!
> shoot A 1
Unfortunately you have missed! Now it's your enemy
's turn!
> shoot A 2
It's not your turn now!
> stats Artem
You have 0 wins and 0 loses!
FULL STATISTICS:
        0 kills
        0 wounds
        1 misses
> 
```

```
moroz0v@LAPTOP-T5JMDNV1:~$ cd OS/cp
moroz0v@LAPTOP-T5JMDNV1:~/OS/cp$ ./client
Welcome to the SeaBattle! Please enter your nickn
ame:
> Nikita
Hello, Nikita!
Follow next rules:
        create for creating a new game
        connect for connecting to the server
        shoot for shooting at enemy's ship
        stats for checking your stats
        disconnect for leaving from the server
        quit for leaving from the program
        help for checking rules
> connect game 12345
Connected sucessfully
You are a player №2, cause you have connected to
the game. Your field has been prepared!
> shoot A 1
Unfortunately you have missed! Now it's your enem
y's turn!
> shoot A 2
It's not your turn now!
> stats Nikita
You have 0 wins and 0 loses!
FULL STATISTICS:
        0 kills
        0 wounds
        1 misses
> 
```

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, а также впервые в своей жизни написал клиент-серверную игру.