МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Курсовой проект
по курсу «Операционные системы»

Выполнил: Д. А. Алгиничев
Группа: М8О-208БВ-24
Преподаватель: Е. С. Миронов
Дата сдачи: 15.12.2025

Москва, 2025

Условие

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

• Создать игру, введя ее имя

• Присоединиться к одной из существующих игр по имени игры

Цель работы

1. Приобретение практических навыков в использовании знаний, полученных в течении курса

2. Проведение исследования в выбранной предметной области

Задание

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

Задание варианта

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

Вариант

6

Архитектура проекта

Архитектура проекта реализует клиент-серверную модель с использованием разделяемой памяти (shared memory) для межпроцессного взаимодействия. Система состоит из трёх ключевых компонентов, определённых в файлах проекта:

- **Сервер** (`Server.hpp`, `Server.cpp`) — центральный координатор, который управляет всеми игровыми сессиями, обрабатывает команды клиентов и поддерживает целостность игрового состояния. Сервер работает в бесконечном цикле, ожидая сообщений в общей очереди.
- **Клиенты** (`Client.hpp`, `Client.cpp`) — независимые процессы, предоставляющие консольный интерфейс игрокам. Каждый клиент подключается к существующей разделяемой памяти, регистрируется с уникальным логином и взаимодействует с системой через меню.

- **Разделяемая память** (SharedTypes.hpp, SharedMemory.hpp, SharedMemory.cpp) — общая область памяти, содержащая структуру SharedMemoryRoot с очередью сообщений, слотами клиентов и массивами игр. Этот компонент обеспечивает высокоскоростное взаимодействие между процессами.

Все компоненты взаимодействуют через единую структуру данных, определённую в SharedTypes.hpp, которая включает типы сообщений (MsgType), состояния ячеек (CellState), кораблей (Ship) и игр (GameData). Синхронизация осуществляется с помощью POSIX мьютексов и условных переменных, также определённых в структуре разделяемой памяти.

Метод решения

Метод решения основан на использовании POSIX Shared Memory для реализации межпроцессного взаимодействия между сервером и клиентами. В файле SharedTypes.hpp определена центральная структура SharedMemoryRoot, содержащая:

- **Очередь сообщений** (Message queue[QUEUE_SIZE]) — циклический буфер для асинхронной передачи команд от клиентов серверу и обратно. Типы сообщений определены в перечислении MsgType и включают все возможные действия: регистрация (MSG_REGISTER), создание игры (MSG_CREATE), выстрел (MSG_SHOT) и др.
- **Слоты клиентов** (ClientSlot clients[MAX_CLIENTS]) — информация о каждом подключённом игроке, включая логин, условную переменную для ожидания ответов и текущий статус.
- **Игровые сессии** (GameData games[16]) — полное состояние каждой игры, включая поля игроков (board1, board2), массивы кораблей (ships1, ships2), статистику и временные метки.

Серверный процесс (реализованный в Server.cpp) работает по следующему алгоритму:

1. Инициализирует разделяемую память и создаёт необходимые объекты синхронизации.
2. Входит в основной цикл, ожидая сообщений через условную переменную server_cond.
3. При получении сообщения определяет его тип и вызывает соответствующий обработчик (handle_message()).
4. Обновляет состояние игр, отправляет ответы клиентам и синхронизирует изменения.

Клиентский процесс (Client.cpp) реализует:

1. Подключение к существующей разделяемой памяти.
2. Регистрацию с уникальным логином через сообщение MSG_REGISTER.
3. Взаимодействие с пользователем через контекстные меню (главное меню, меню расстановки, игровое меню).

4. Отправку команд в очередь и ожидание ответов через условные переменные.

Игровая логика инкапсулирована в классе `Game` (`Game.hpp`, `Game.cpp`), который отвечает за:

- Валидацию расстановки кораблей (проверка границ, пересечений, расстояний).
- Обработку выстрелов с определением попаданий, промахов и уничтожения кораблей.
- Управление очерёдностью ходов (правило дополнительного хода при попадании).
- Определение условия победы (уничтожение всех кораблей противника).

Синхронизация обеспечивается через мьютекс `mutex` в структуре `SharedMemoryRoot`. Каждая операция чтения или записи в разделяемую память предваряется захватом этого мьютекса, что предотвращает состояния гонки (race conditions). Условные переменные (`server_cond` и `cond` в каждом `ClientSlot`) используются для эффективного ожидания событий без активного опроса.

Обработка ошибок реализована на всех уровнях: проверка корректности входных данных (координаты в пределах 0-9, правильный формат команд), валидация игровых действий (очередность хода, завершённость расстановки), обработка исключительных ситуаций (отключение клиента, переполнение очереди).

Описание программы

Сервер (server)

Сервер запускается первым и выполняет следующие функции:

- Создаёт и инициализирует разделяемую память (если не существует).
- Устанавливает структуры данных: очередь сообщений, слоты клиентов, игровые сессии.
- Инициализирует мьютексы и условные переменные с атрибутами `PTHREAD_PROCESS_SHARED`.
- Ожидает команды от клиентов в бесконечном цикле, обрабатывая их в порядке поступления.
- Управляет жизненным циклом игр: создание, наполнение игроками, проведение игры, завершение и очистка.
- Обеспечивает целостность данных и соблюдение правил игры.

Клиент (client)

Клиентское приложение предоставляет пользователю консольный интерфейс с контекстными меню:

1. **Главное меню** (отображается когда игрок не в игре):

- Список игроков и игр (команда `list` → `MSG_LIST`)
- Создание публичной игры (ввод имени → `MSG_CREATE`)
- Присоединение к игре по имени или ID (`MSG_JOIN`)
- Приглашение другого игрока по логину (`MSG_INVITE`)
- Проверка приглашений
- Выход из системы (`MSG_QUIT`)

2. **Меню расстановки кораблей** (после входа в игру):
   - Ручное размещение: `place размер,x,y,ориентация` → `MSG_PLACE_SHIP`
   - Автоматическая расстановка: `auto` (алгоритм в `Client.cpp`)
   - Просмотр своего поля: `board` → `MSG_GET_BOARD`
   - Завершение расстановки: `ready` → `MSG_SETUP_COMPLETE`
   - Приглашение в текущую игру: `invite логин` → `MSG_INVITE_TO_GAME`
   - Выход из игры: `menu` → `MSG_LEAVE_GAME`
3. **Игровое меню** (после начала боя):
   - Сделать выстрел: ввод координат → `MSG_SHOT`
   - Просмотр своего поля → `MSG_GET_BOARD`
   - Просмотр поля противника → `MSG_GET_OPPONENT_BOARD`
   - Статус игры → `MSG_GAME_STATUS`
   - Сдаться → `MSG_SURRENDER`
   - Выйти в меню → `MSG_LEAVE_GAME`

Игровой процесс

Игра следует классическим правилам "Морского боя":

- Поле 10×10 клеток.
- Флот из 10 кораблей: 1×4, 2×3, 3×2, 4×1 клетки.
- Расстановка с расстоянием минимум 1 клетка между кораблями.
- Поочерёдные ходы с правом дополнительного хода при попадании.
- Отображение полей: своё поле показывает корабли, поле противника скрывает неподбитые корабли.
- Победа при уничтожении всех кораблей противника.

Особенности реализации

- **Автоматическая расстановка** использует алгоритм случайного размещения с проверкой корректности позиции.
- **Система приглашений** поддерживает два сценария: создание новой приватной игры и приглашение в существующую игру.
- **Визуализация** полей с помощью символов ASCII: `.` — пусто, `S` — корабль (на своём поле), `X` — попадание, `O` — промах, `#` — потопленный корабль.
- **Обработка отключений**: сервер обнаруживает неактивных клиентов и корректно завершает игры.
- **Масштабирование**: ограничение на 32 одновременных клиента и 16 активных игр (константы в `SharedTypes.hpp`).

Пример работы программы

Запуск сервера

Первым запускается сервер. Он создаёт разделяемую память и инициализирует все структуры данных:

```
$ ./server
=== SERVER RUNNING ===
Server: shared memory initialized
```

Сервер готов к приёму подключений клиентов и ожидает сообщений в бесконечном цикле.

Запуск клиентов и регистрация

Запускаем двух клиентов в разных терминалах. Первый игрок регистрируется как "Alice":

```
$ ./client
==================================================
ДОБРО ПОЖАЛОВАТЬ В МОРСКОЙ БОЙ!
==================================================
Введите ваш логин: Alice

[Регистрация...]

REGISTERED:OK
```

Второй игрок регистрируется как "Bob":

```
$ ./client
==================================================
ДОБРО ПОЖАЛОВАТЬ В МОРСКОЙ БОЙ!
==================================================
```

```
Введите ваш логин: Bob

[Регистрация...]

REGISTERED:OK
```

На сервере появляются соответствующие сообщения:

```
Registered client: Alice
Registered client: Bob
```

Создание и поиск игры

Alice создаёт публичную игру "Battle1":

```
==================================================
МОРСКОЙ БОЙ
==================================================
1 -Список игроков и игр
2 -Создать публичную игру
3 -Присоединиться к игре
4 -Пригласить игрока
5 -Проверить приглашения
6 -Выйти
Выберите действие: 2

Введите имя для новой игры: Battle1

GAME_CREATED:Публичная игра 'Battle1'создана (ID: 0)
Вы вошли в игру. Начинайте расстановку кораблей!
```

На сервере:

```
Public game 'Battle1'created by Alice (ID: 0)
```

Bob запрашивает список доступных игр:

```
==================================================
МОРСКОЙ БОЙ
==================================================
1 -Список игроков и игр
2 -Создать публичную игру
3 -Присоединиться к игре
4 -Пригласить игрока
5 -Проверить приглашения
6 -Выйти
Выберите действие: 1
```

```
=== ИГРОКИ ОНЛАЙН (2) ===
Alice [в игре]
Bob

=== ДОСТУПНЫЕ ИГРЫ (1) ===
Battle1 (ID: 0) -создатель: Alice

Присоединиться: join <имя_игры>или join <ID>
```

Bob присоединяется к игре по ID:

```
Выберите действие: 3

Введите имя или ID игры: 0

Вы присоединились к игре!
SHIP_PLACEMENT:
Разместите корабли: place размер,x,y,ориентация(H/V)
Корабли: 1x4,2x3,3x2,4x1
Пример: place 4,0,0,H
Когда готовы: ready
```

Alice получает уведомление:

```
OPPONENT_JOINED:Игрок присоединился
SHIP_PLACEMENT:
Разместите корабли: place размер,x,y,ориентация(H/V)
Корабли: 1x4,2x3,3x2,4x1
Пример: place 4,0,0,H
Когда готовы: ready
```

Фаза расстановки кораблей

Оба игрока начинают расставлять корабли. Alice использует автоматическую расстановку:

```
==================================================
РАССТАНОВКА КОРАБЛЕЙ
==================================================
Формат: размер,x,y,ориентация(H/V)
Пример: 4,0,0,H

Корабли для размещения:
1 авианосец (4 клеток)
2 линкора (3 клетки)
3 крейсера (2 клетки)
4 эсминца (1 клетки)
--------------------------------------------------
Команды:
```

```
auto -автоматическая расстановка
ready -готов к игре
board -посмотреть поле
invite <логин>-пригласить игрока в эту игру
menu -выйти в меню
-------------------------------------------------

Команда: auto

=================================================
АВТОМАТИЧЕСКАЯ РАССТАНОВКА КОРАБЛЕЙ
=================================================
Размещаем корабль: 4,0,0,H... Успешно
Размещаем корабль: 3,0,5,H... Успешно
Размещаем корабль: 3,0,8,H... Успешно
Размещаем корабль: 2,8,8,H... Успешно
Размещаем корабль: 2,5,5,V... Успешно
Размещаем корабль: 2,8,4,H... Успешно
Размещаем корабль: 1,6,0,H... Успешно
Размещаем корабль: 1,6,2,H... Успешно
Размещаем корабль: 1,3,2,H... Успешно
Размещаем корабль: 1,9,0,H... Успешно
-------------------------------------------------
РАССТАНОВКА ЗАВЕРШЕНА
Размещено кораблей: 10/10
Все корабли успешно размещены!

Показываем поле...
```

Alice смотрит своё поле:

```
Команда: board

YOUR_BOARD:
0  1  2  3  4  5  6  7  8  9
0  S  S  S  S  .  .  .  .  .  .
1  .  .  .  .  .  .  .  .  .  .
2  .  .  .  S  .  .  S  .  .  .
3  .  .  .  .  .  .  .  .  .  .
4  .  .  .  .  .  .  .  .  S  S
5  .  .  .  .  .  S  .  .  .  .
6  .  .  .  .  .  S  .  .  .  .
7  .  .  .  .  .  S  .  .  .  .
8  .  .  .  .  .  .  .  .  S  S
9  S  .  .  .  .  .  .  .  .  .
```

Bob размещает корабли вручную:

```
Команда: place 4,0,0,H
```

```
SHIP_PLACED:OK

Команда: place 3,0,5,H
SHIP_PLACED:OK

Команда: place 3,0,8,H
SHIP_PLACED:OK
```

После размещения всех кораблей Bob отправляет команду готовности:

```
Команда: ready

Отправляем 'ready'на сервер...

SETUP_COMPLETE:Waiting for opponent...
```

Когда Alice также отправляет "ready игра переходит в активную фазу:

```
Команда: ready

Отправляем 'ready'на сервер...

ВАШ ХОД! Make your move
```

Фаза боя

Игра начинается. Alice ходит первой:

```
========================================
ИГРА В ПРОЦЕССЕ
========================================
1 -Сделать выстрел
2 -Посмотреть свое поле
3 -Посмотреть поле противника
4 -Статус игры
5 -Сдаться
6 -Выйти в меню
----------------------------------------
Выберите действие: 1

Координаты выстрела (x,y): 5,5

Отправляем выстрел...

Ответ сервера получен

РЕЗУЛЬТАТ ВЫСТРЕЛА: MISS at 5,5
```

Bob получает уведомление и делает свой ход:

```
ПРОТИВНИК СТРЕЛЯЕТ: Alice at 5,5:MISS
ВАШ ХОД! Make your move

Выберите действие: 1

Координаты выстрела (x,y): 0,0

Отправляем выстрел...

РЕЗУЛЬТАТ ВЫСТРЕЛА: HIT at 0,0

ВАШ ХОД СНОВА! You hit! Shoot again
```

Завершение игры

После нескольких ходов Bob уничтожает последний корабль Alice:

```
РЕЗУЛЬТАТ ВЫСТРЕЛА: HIT at 9,0
==================================================
ПОБЕДА!
You won the game!
==================================================

FINAL_STATS:
Статистика:
Сбито кораблей: 10
Попаданий: 15
Промахов: 3
Игра завершена!
Вы победили!
```

Alice получает уведомление о поражении:

```
ПРОТИВНИК СТРЕЛЯЕТ: Bob at 9,0:HIT
==================================================
ПОРАЖЕНИЕ
You lost the game
==================================================

FINAL_STATS:
Статистика:
Сбито кораблей: 7
Попаданий: 10
Промахов: 8
Игра завершена!
Победил: Bob
```

Выход из системы

Игрок может выйти из системы:

```
==================================================
МОРСКОЙ БОЙ
==================================================
1 -Список игроков и игр
2 -Создать публичную игру
3 -Присоединиться к игре
4 -Пригласить игрока
5 -Проверить приглашения
6 -Выйти
Выберите действие: 6

Вы уверены? (да/нет): да

Выход...

==================================================
ИГРА ЗАВЕРШЕНА
Спасибо за игру!
==================================================
```

На сервере фиксируется отключение клиента:

```
Client quit: Alice
```

Выводы

В процессе выполнения курсового проекта я получил практический опыт проектирования и реализации клиент-серверной системы с использованием разделяемой памяти для межпроцессного взаимодействия.

Архитектура на основе shared memory - позволяет достичь высокой производительности за счёт минимальных накладных расходов на передачу данных между процессами. Преимущества: быстрое взаимодействие, простота синхронизации через мьютексы, единое представление данных для всех компонентов. Недостатки: ограничение на количество процессов, сложность масштабирования на несколько машин, требования к корректной синхронизации.

Клиент-серверная модель - обеспечивает четкое разделение ответственности: сервер как координатор и валидатор, клиенты как пользовательские интерфейсы. Преимущества: централизованное управление состоянием, безопасность (проверка действий на сервере), возможность добавления новых клиентов без изменения сервера. Недостатки: единая точка отказа, зависимость от доступности сервера.

Приобретенные навыки

- **Работа с POSIX shared memory** - освоил механизмы создания, отображения и управления разделяемой памятью в Linux

- **Межпроцессная синхронизация** - изучил использование мьютексов и условных переменных в многопроцессной среде
- **Проектирование протоколов взаимодействия** - разработал систему сообщений для клиент-серверного общения
- **Игровая логика и бизнес-правила** - реализовал полный цикл игры "Морской бой"с соблюдением всех правил
- **Консольный пользовательский интерфейс** - создал интуитивно понятные меню для взаимодействия с игроком

Заключение

Использование разделяемой памяти для реализации многопользовательской игры доказало свою эффективность для локальных систем. Такой подход особенно полезен в случаях, когда требуется:

- Высокая производительность при интенсивном обмене данными
- Простота отладки и мониторинга (все данные в одном месте)
- Минимальные задержки между клиентами и сервером

Полученные знания и навыки позволяют создавать высокопроизводительные многопользовательские приложения с четкой архитектурой и надежной синхронизацией. Проект демонстрирует, что даже классические задачи (такие как "Морской бой") могут быть реализованы с использованием современных технологий межпроцессного взаимодействия, что открывает возможности для создания более сложных распределенных систем в будущем. Особенно ценным оказался опыт работы с реальными проблемами синхронизации, обработки отказов и проектирования отказоустойчивых систем. Эти компетенции критически важны для разработки промышленного программного обеспечения.

Исходная программа

```
#pragma once
#include <pthread.h>
#include <cstdint>
#include <cstring>

constexpr const char* SHM_NAME = "/battleship_shm_v3";
constexpr size_t MAX_CLIENTS = 32;
constexpr size_t QUEUE_SIZE = 128;
constexpr size_t LOGIN_MAX = 32;
constexpr size_t CMD_MAX = 256;
constexpr size_t RESP_MAX = 512;

constexpr int BOARD_SIZE = 10;
constexpr int MAX_SHIPS = 10;

enum CellState : uint8_t {
```

```
17      CELL_EMPTY = 0,
18      CELL_SHIP = 1,
19      CELL_HIT = 2,
20      CELL_MISS = 3,
21      CELL_SUNK = 4
22  };
23
24  enum ShipType : uint8_t {
25      SHIP_CARRIER = 4,
26      SHIP_BATTLESHIP = 3,
27      SHIP_CRUISER = 2,
28      SHIP_DESTROYER = 1
29  };
30
31  enum GameState : uint8_t {
32      GAME_WAITING = 0,
33      GAME_SETUP = 1,
34      GAME_ACTIVE = 2,
35      GAME_FINISHED = 3
36  };
37
38  struct Ship {
39      uint8_t size;
40      uint8_t health;
41      bool horizontal;
42      uint8_t start_x;
43      uint8_t start_y;
44      bool sunk;
45  };
46
47  struct GameData {
48      bool used;
49      char game_name[LOGIN_MAX];
50      char player1[LOGIN_MAX];
51      char player2[LOGIN_MAX];
52      GameState state;
53      char current_turn[LOGIN_MAX];
54      bool is_public;
55
56      CellState board1[BOARD_SIZE][BOARD_SIZE];
57      CellState board2[BOARD_SIZE][BOARD_SIZE];
58
59      Ship ships1[MAX_SHIPS];
60      Ship ships2[MAX_SHIPS];
61      uint8_t ship_count1;
62      uint8_t ship_count2;
63
64      uint8_t hits1;
65      uint8_t hits2;
66      uint8_t misses1;
67      uint8_t misses2;
```

```
68      uint8_t sunk1;
69      uint8_t sunk2;
70
71      time_t start_time;
72      time_t end_time;
73  };
74
75  enum MsgType : uint8_t {
76      MSG_REGISTER = 1,
77      MSG_LIST = 2,
78      MSG_INVITE = 3,
79      MSG_INVITE_TO_GAME = 16,
80      MSG_ACCEPT = 4,
81      MSG_SHOT = 5,
82      MSG_QUIT = 6,
83      MSG_SETUP_COMPLETE = 7,
84      MSG_PLACE_SHIP = 8,
85      MSG_GET_BOARD = 9,
86      MSG_GET_OPPONENT_BOARD = 10,
87      MSG_SURRENDER = 11,
88      MSG_GAME_STATUS = 12,
89      MSG_CREATE = 13,
90      MSG_JOIN = 14,
91      MSG_LEAVE_GAME = 15
92  };
93
94  struct Message {
95      bool used;
96      char from[LOGIN_MAX];
97      char to[LOGIN_MAX];
98      uint8_t type;
99      char payload[CMD_MAX];
100 };
101
102 struct ClientSlot {
103     bool used;
104     char login[LOGIN_MAX];
105     pthread_cond_t cond;
106     char response[RESP_MAX];
107     bool has_response;
108     int current_game_id;
109     bool setup_complete;
110 };
111
112 struct SharedMemoryRoot {
113     pthread_mutex_t mutex;
114     pthread_cond_t server_cond;
115
116     Message queue[QUEUE_SIZE];
117     size_t q_head;
118     size_t q_tail;
```

```
119
120     ClientSlot clients[MAX_CLIENTS];
121
122     GameData games[16];
123     size_t game_count;
124 };
```

Листинг 1: SharedTypes.hpp - определения типов данных и структур для разделяемой памяти

```
1  #pragma once
2  #include "SharedTypes.hpp"
3  #include <sys/mman.h>
4  #include <sys/stat.h>
5  #include <fcntl.h>
6  #include <unistd.h>
7  #include <stdexcept>
8  #include <cstring>
9  #include <string>
10 #include <iostream>
11
12 class SharedMemory {
13 public:
14     SharedMemory(bool create = false);
15     ~SharedMemory();
16
17     SharedMemoryRoot* root() { return _root; }
18     bool is_owner() const { return owner; }
19
20 private:
21     int fd;
22     SharedMemoryRoot* _root;
23     bool owner;
24 };
```

Листинг 2: SharedMemory.hpp - интерфейс класса для работы с разделяемой памятью

```
1  #include "SharedMemory.hpp"
2  #include <iostream>
3
4  SharedMemory::SharedMemory(bool create)
5      : fd(-1), _root(nullptr), owner(false)
6  {
7      bool do_create = create;
8      if (do_create) {
9          fd = shm_open(SHM_NAME, O_CREAT | O_RDWR, 0666);
10         if (fd < 0) throw std::runtime_error("shm_open create failed");
11         if (ftruncate(fd, sizeof(SharedMemoryRoot)) != 0) {
12             close(fd);
```

```
            throw std::runtime_error("ftruncate failed");
        }
        owner = true;
    } else {
        fd = shm_open(SHM_NAME, O_RDWR, 0666);
        if (fd < 0) throw std::runtime_error("shm_open open failed; run server
            first");
    }
    void* addr = mmap(nullptr, sizeof(SharedMemoryRoot), PROT_READ | PROT_WRITE,
         MAP_SHARED, fd, 0);
    if (addr == MAP_FAILED) {
        close(fd);
        throw std::runtime_error("mmap failed");
    }
    _root = reinterpret_cast<SharedMemoryRoot*>(addr);
}

SharedMemory::~SharedMemory() {
    if (_root) munmap(_root, sizeof(SharedMemoryRoot));
    if (fd >= 0) close(fd);
    if (owner) {
        shm_unlink(SHM_NAME);
    }
}
```

Листинг 3: SharedMemory.cpp - реализация работы с разделяемой памятью

```
#pragma once
#include "../include/SharedTypes.hpp"
#include "../include/SharedMemory.hpp"
#include "Game.hpp"
#include <string>
#include <vector>
#include <unordered_map>

class Server {
public:
    Server();
    ~Server();
    void run();

private:
    SharedMemory shm;
    SharedMemoryRoot* root;
    bool setup_done;

    std::unordered_map<int, Game*> games_map;

    void init_shared_objects();
    void handle_message(const Message &m);
```

```
24      void send_response_to(const char* login, const char* text);
25
26      ClientSlot* find_or_create_client(const char* login);
27      ClientSlot* find_client(const char* login);
28      std::vector<std::string> list_clients();
29      std::vector<std::string> list_available_games();
30
31      int create_private_game(const std::string& creator, const std::string&
            target);
32      int create_public_game(const std::string& game_name, const std::string&
            creator);
33      Game* find_game_by_name(const std::string& game_name);
34      Game* get_game(int game_id);
35      void remove_game(int game_id);
36
37      void handle_setup_complete(const Message &m);
38      void handle_place_ship(const Message &m);
39      void handle_get_board(const Message &m);
40      void handle_get_opponent_board(const Message &m);
41      void handle_surrender(const Message &m);
42      void handle_game_status(const Message &m);
43
44      bool parse_ship_placement(const std::string& payload, uint8_t& size,
45                                uint8_t& x, uint8_t& y, bool& horizontal);
46      bool parse_shot(const std::string& payload, uint8_t& x, uint8_t& y);
47  };
```

Листинг 4: Server.hpp - интерфейс сервера игры "Морской бой"

```
1   #include "Server.hpp"
2
3   #include <algorithm>
4   #include <cstring>
5   #include <iostream>
6   #include <sstream>
7
8   Server::Server() : shm(true), root(shm.root()), setup_done(false) {
9       if (shm.is_owner()) {
10          init_shared_objects();
11      }
12  }
13
14  Server::~Server() {
15      for (auto& pair : games_map) {
16          delete pair.second;
17      }
18  }
19
20  void Server::init_shared_objects() {
21      pthread_mutexattr_t mattr;
```

```
22      pthread_condattr_t cattr;
23      pthread_mutexattr_init(&mattr);
24      pthread_mutexattr_setpshared(&mattr, PTHREAD_PROCESS_SHARED);
25      pthread_condattr_init(&cattr);
26      pthread_condattr_setpshared(&cattr, PTHREAD_PROCESS_SHARED);
27
28      pthread_mutex_init(&root->mutex, &mattr);
29      pthread_cond_init(&root->server_cond, &cattr);
30
31      root->q_head = root->q_tail = 0;
32      root->game_count = 0;
33
34      for (size_t i = 0; i < QUEUE_SIZE; ++i)
35          root->queue[i].used = false;
36      for (size_t i = 0; i < MAX_CLIENTS; ++i) {
37          root->clients[i].used = false;
38          root->clients[i].has_response = false;
39          root->clients[i].current_game_id = -1;
40          root->clients[i].setup_complete = false;
41          pthread_cond_init(&root->clients[i].cond, &cattr);
42          std::memset(root->clients[i].login, 0, sizeof(root->clients[i].login));
43          std::memset(root->clients[i].response, 0, sizeof(root->clients[i].
                response));
44      }
45
46      for (size_t i = 0; i < 16; ++i) {
47          root->games[i].used = false;
48      }
49
50      pthread_mutexattr_destroy(&mattr);
51      pthread_condattr_destroy(&cattr);
52
53      setup_done = true;
54      std::cout << "Server: shared memory initialized\n";
55  }
56
57  ClientSlot* Server::find_or_create_client(const char* login) {
58      for (size_t i = 0; i < MAX_CLIENTS; ++i) {
59          if (root->clients[i].used && std::strncmp(root->clients[i].login, login,
                LOGIN_MAX) == 0) {
60              return &root->clients[i];
61          }
62      }
63      for (size_t i = 0; i < MAX_CLIENTS; ++i) {
64          if (!root->clients[i].used) {
65              root->clients[i].used = true;
66              std::strncpy(root->clients[i].login, login, LOGIN_MAX - 1);
67              root->clients[i].has_response = false;
68              root->clients[i].current_game_id = -1;
69              root->clients[i].setup_complete = false;
70              std::memset(root->clients[i].response, 0, RESP_MAX);
```

```
71              return &root->clients[i];
72          }
73      }
74      return nullptr;
75  }
76
77  ClientSlot* Server::find_client(const char* login) {
78      for (size_t i = 0; i < MAX_CLIENTS; ++i) {
79          if (root->clients[i].used && std::strncmp(root->clients[i].login, login,
                LOGIN_MAX) == 0) {
80              return &root->clients[i];
81          }
82      }
83      return nullptr;
84  }
85
86  std::vector<std::string> Server::list_clients() {
87      std::vector<std::string> res;
88      for (size_t i = 0; i < MAX_CLIENTS; ++i) {
89          if (root->clients[i].used) {
90              std::string info = root->clients[i].login;
91              if (root->clients[i].current_game_id != -1) {
92                  info += " [в игре]";
93              }
94              res.emplace_back(info);
95          }
96      }
97      return res;
98  }
99
100 std::vector<std::string> Server::list_available_games() {
101     std::vector<std::string> res;
102     for (int i = 0; i < 16; i++) {
103         if (root->games[i].used && root->games[i].is_public &&
104             root->games[i].state == GAME_WAITING) {
105             std::string info = " " + std::string(root->games[i].game_name) +
106                                " (ID: " + std::to_string(i) +
107                                ") - создатель: " + std::string(root->games[i].
                                    player1);
108             res.emplace_back(info);
109         }
110     }
111     return res;
112 }
113
114 void Server::send_response_to(const char* login, const char* text) {
115     pthread_mutex_lock(&root->mutex);
116     ClientSlot* cl = find_client(login);
117     if (cl) {
118         std::strncpy(cl->response, text, RESP_MAX - 1);
119         cl->has_response = true;
```

```
120         pthread_cond_signal(&cl->cond);
121     }
122     pthread_mutex_unlock(&root->mutex);
123 }
124 
125 int Server::create_private_game(const std::string& creator, const std::string&
        target) {
126     if (root->game_count >= 16)
127         return -1;
128 
129     std::string game_name = creator + "_vs_" + target;
130     int game_id = -1;
131 
132     for (int i = 0; i < 16; i++) {
133         if (!root->games[i].used) {
134             game_id = i;
135             break;
136         }
137     }
138 
139     if (game_id == -1)
140         return -1;
141 
142     Game* game = new Game(game_name, creator, root, false);
143     games_map[game_id] = game;
144     root->game_count++;
145 
146     ClientSlot* client = find_client(creator.c_str());
147     if (client) {
148         client->current_game_id = game_id;
149     }
150 
151     std::cout << "Private game '" << game_name << "' created by " << creator <<
            " for " << target
152               << std::endl;
153     return game_id;
154 }
155 
156 int Server::create_public_game(const std::string& game_name, const std::string&
        creator) {
157     if (root->game_count >= 16)
158         return -1;
159 
160     for (int i = 0; i < 16; i++) {
161         if (root->games[i].used && std::strcmp(root->games[i].game_name,
                game_name.c_str()) == 0) {
162             return -2;
163         }
164     }
165 
166     int game_id = -1;
```

```
167     for (int i = 0; i < 16; i++) {
168         if (!root->games[i].used) {
169             game_id = i;
170             break;
171         }
172     }
173 
174     if (game_id == -1)
175         return -1;
176 
177     // Создаемигру
178     Game* game = new Game(game_name, creator, root, true);
179     games_map[game_id] = game;
180     root->game_count++;
181 
182     ClientSlot* client = find_client(creator.c_str());
183     if (client) {
184         client->current_game_id = game_id;
185         client->setup_complete = false;
186     }
187 
188     std::cout << "Public game '" << game_name << "' created by " << creator << "
          (ID: " << game_id
189               << ")" << std::endl;
190     return game_id;
191 }
192 
193 Game* Server::find_game_by_name(const std::string& game_name) {
194     for (auto& pair : games_map) {
195         if (pair.second->get_game_name() == game_name) {
196             return pair.second;
197         }
198     }
199     return nullptr;
200 }
201 
202 Game* Server::get_game(int game_id) {
203     auto it = games_map.find(game_id);
204     if (it != games_map.end()) {
205         return it->second;
206     }
207     return nullptr;
208 }
209 
210 void Server::remove_game(int game_id) {
211     auto it = games_map.find(game_id);
212     if (it != games_map.end()) {
213         Game* game = it->second;
214 
215         std::string player1 = game->get_player1();
216         std::string player2 = game->get_player2();
```

```
217
218         if (!player1.empty()) {
219             ClientSlot* client1 = find_client(player1.c_str());
220             if (client1) {
221                 client1->current_game_id = -1;
222                 client1->setup_complete = false;
223                 send_response_to(player1.c_str(), "GAME_REMOVED:Игра удалена");
224             }
225         }
226
227         if (!player2.empty()) {
228             ClientSlot* client2 = find_client(player2.c_str());
229             if (client2) {
230                 client2->current_game_id = -1;
231                 client2->setup_complete = false;
232                 send_response_to(player2.c_str(), "GAME_REMOVED:Игра удалена");
233             }
234         }
235
236         delete game;
237         games_map.erase(it);
238         root->game_count--;
239
240         if (game_id >= 0 && game_id < 16) {
241             root->games[game_id].used = false;
242         }
243     }
244 }
245
246 bool Server::parse_ship_placement(const std::string& payload, uint8_t& size,
    uint8_t& x, uint8_t& y,
247                                   bool& horizontal) {
248     int s, x_pos, y_pos;
249     char orientation;
250
251     if (sscanf(payload.c_str(), "%d,%d,%d,%c", &s, &x_pos, &y_pos, &orientation)
         != 4) {
252         return false;
253     }
254
255     if (s < 1 || s > 4)
256         return false;
257     if (x_pos < 0 || x_pos >= BOARD_SIZE)
258         return false;
259     if (y_pos < 0 || y_pos >= BOARD_SIZE)
260         return false;
261     if (orientation != 'H' && orientation != 'V')
262         return false;
263
264     size = static_cast<uint8_t>(s);
265     x = static_cast<uint8_t>(x_pos);
```

```
266     y = static_cast<uint8_t>(y_pos);
267     horizontal = (orientation == 'H');
268 
269     return true;
270 }
271 
272 bool Server::parse_shot(const std::string& payload, uint8_t& x, uint8_t& y) {
273     int x_pos, y_pos;
274 
275     if (sscanf(payload.c_str(), "%d,%d", &x_pos, &y_pos) != 2) {
276         return false;
277     }
278 
279     if (x_pos < 0 || x_pos >= BOARD_SIZE)
280         return false;
281     if (y_pos < 0 || y_pos >= BOARD_SIZE)
282         return false;
283 
284     x = static_cast<uint8_t>(x_pos);
285     y = static_cast<uint8_t>(y_pos);
286 
287     return true;
288 }
289 
290 void Server::handle_setup_complete(const Message& m) {
291     ClientSlot* client = find_client(m.from);
292     if (!client) {
293         send_response_to(m.from, "ERROR:Not registered");
294         return;
295     }
296 
297     if (client->current_game_id == -1) {
298         send_response_to(m.from, "ERROR:Not in a game");
299         return;
300     }
301 
302     Game* game = get_game(client->current_game_id);
303     if (!game) {
304         send_response_to(m.from, "ERROR:Game not found");
305         return;
306     }
307 
308     if (!game->is_setup_complete(m.from)) {
309         send_response_to(m.from, "SETUP_INCOMPLETE:Place all ships first");
310         return;
311     }
312 
313     client->setup_complete = true;
314     game->set_setup_complete(m.from);
315 
316     send_response_to(m.from, "SETUP_COMPLETE:Waiting for opponent...");
```

```
317 }
318
319 void Server::handle_place_ship(const Message& m) {
320     ClientSlot* client = find_client(m.from);
321     if (!client) {
322         send_response_to(m.from, "ERROR:Not registered");
323         return;
324     }
325
326     if (client->current_game_id == -1) {
327         send_response_to(m.from, "ERROR:Not in a game");
328         return;
329     }
330
331     Game* game = get_game(client->current_game_id);
332     if (!game) {
333         send_response_to(m.from, "ERROR:Game not found");
334         return;
335     }
336
337     if (game->is_game_active() || game->is_game_finished()) {
338         send_response_to(m.from, "ERROR:Game already started or finished");
339         return;
340     }
341
342     uint8_t size, x, y;
343     bool horizontal;
344
345     if (!parse_ship_placement(m.payload, size, x, y, horizontal)) {
346         send_response_to(m.from, "SHIP_ERROR:Invalid format. Use: size,x,y,
            orientation(H/V)");
347         return;
348     }
349
350     if (game->place_ship(m.from, size, x, y, horizontal)) {
351         send_response_to(m.from, "SHIP_PLACED:OK");
352
353         std::string board = game->get_player_board(m.from, true);
354         send_response_to(m.from, ("YOUR_BOARD:\n" + board).c_str());
355
356         if (game->is_setup_complete(m.from)) {
357             send_response_to(m.from, "ALL_SHIPS_PLACED:Use 'ready' when done");
358         }
359     } else {
360         send_response_to(m.from, "SHIP_ERROR:Cannot place ship here");
361     }
362 }
363
364 void Server::handle_get_board(const Message& m) {
365     ClientSlot* client = find_client(m.from);
366     if (!client || client->current_game_id == -1) {
```

```
367         send_response_to(m.from, "ERROR:Not in a game");
368         return;
369     }
370 
371     Game* game = get_game(client->current_game_id);
372     if (!game) {
373         send_response_to(m.from, "ERROR:Game not found");
374         return;
375     }
376 
377     std::string board = game->get_player_board(m.from, true);
378     send_response_to(m.from, ("YOUR_BOARD:\n" + board).c_str());
379 }
380 
381 void Server::handle_get_opponent_board(const Message& m) {
382     ClientSlot* client = find_client(m.from);
383     if (!client || client->current_game_id == -1) {
384         send_response_to(m.from, "ERROR:Not in a game");
385         return;
386     }
387 
388     Game* game = get_game(client->current_game_id);
389     if (!game) {
390         send_response_to(m.from, "ERROR:Game not found");
391         return;
392     }
393 
394     if (!game->is_game_active() && !game->is_game_finished()) {
395         send_response_to(m.from, "ERROR:Game not started yet");
396         return;
397     }
398 
399     std::string board = game->get_opponent_view(m.from);
400     send_response_to(m.from, ("OPPONENT_VIEW:\n" + board).c_str());
401 }
402 
403 void Server::handle_surrender(const Message& m) {
404     ClientSlot* client = find_client(m.from);
405     if (!client || client->current_game_id == -1) {
406         send_response_to(m.from, "ERROR:Not in a game");
407         return;
408     }
409 
410     Game* game = get_game(client->current_game_id);
411     if (!game) {
412         send_response_to(m.from, "ERROR:Game not found");
413         return;
414     }
415 
416     if (game->is_game_finished()) {
417         send_response_to(m.from, "ERROR:Game already finished");
```

```
418         return;
419     }
420 
421     std::string opponent =
422         (game->get_player1() == m.from) ? game->get_player2() : game->
                get_player1();
423 
424     send_response_to(m.from, "SURRENDER:You surrendered");
425     send_response_to(opponent.c_str(), "OPPONENT_SURRENDERED:You win!");
426 
427     remove_game(client->current_game_id);
428 }
429 
430 void Server::handle_game_status(const Message& m) {
431     ClientSlot* client = find_client(m.from);
432     if (!client || client->current_game_id == -1) {
433         send_response_to(m.from, "STATUS:Not in a game");
434         return;
435     }
436 
437     Game* game = get_game(client->current_game_id);
438     if (!game) {
439         send_response_to(m.from, "ERROR:Game not found");
440         return;
441     }
442 
443     std::string status = game->get_status();
444     std::string stats = game->get_statistics(m.from);
445 
446     send_response_to(m.from, ("GAME_STATUS:\n" + status + "\n" + stats).c_str())
            ;
447 }
448 
449 void Server::handle_message(const Message& m) {
450     std::string from(m.from);
451 
452     switch (m.type) {
453     case MSG_REGISTER: {
454         ClientSlot* c = find_or_create_client(m.from);
455         if (c) {
456             send_response_to(m.from, "REGISTERED:OK");
457             std::cout << "Registered client: " << m.from << '\n';
458         } else {
459             send_response_to(m.from, "REGISTERED:FAIL_FULL");
460         }
461         break;
462     }
463     case MSG_LIST: {
464         auto cl = list_clients();
465         auto games = list_available_games();
466 
```

```
467         std::ostringstream ss;
468         ss << "=== ИГРОКИОНЛАЙН(" << cl.size() << ") ===\n";
469         for (auto& n : cl) {
470             ss << " " << n << "\n";
471         }
472 
473         if (!games.empty()) {
474             ss << "\n=== ДОСТУПНЫЕИГРЫ(" << games.size() << ") ===\n";
475             for (auto& g : games) {
476                 ss << " " << g << "\n";
477             }
478             ss << "\nПрисоединиться: join <имя_игры> илиjoin <ID>\n";
479         } else {
480             ss << "\n=== НЕТДОСТУПНЫХИГР===\n";
481             ss << " Создайтеигру: create <имя_игры>\n";
482             ss << " Илипригласите: invite <логин>\n";
483         }
484 
485         send_response_to(m.from, ss.str().c_str());
486         break;
487     }
488     case MSG_INVITE: {
489         const char* target = m.payload;
490         ClientSlot* tgt = find_client(target);
491         ClientSlot* sender = find_client(m.from);
492 
493         if (!tgt) {
494             send_response_to(m.from, "INVITE_FAIL:Игрок ненайден");
495         } else if (tgt->current_game_id != -1) {
496             send_response_to(m.from, "INVITE_FAIL:Игрок ужевигре");
497         } else if (sender->current_game_id != -1) {
498             send_response_to(m.from, "INVITE_FAIL:Вы ужевигре");
499         } else {
500             int game_id = create_private_game(m.from, target);
501             if (game_id == -1) {
502                 send_response_to(m.from, "INVITE_FAIL:Сервер переполнен");
503             } else {
504                 Game* game = get_game(game_id);
505                 if (game) {
506                     char buf[RESP_MAX];
507                     std::snprintf(buf, RESP_MAX, "INVITE:%s:%s:%d", m.from,
508                                   game->get_game_name().c_str(), game_id);
509 
510                     std::cout << " Sending invitation from menu: " << buf << std::
                        endl;
511                     send_response_to(target, buf);
512                     send_response_to(m.from, "INVITE_SENT:Приглашение отправлено")
                        ;
513 
514                     sender->current_game_id = game_id;
515                     sender->setup_complete = false;
```

```
516
517                     std::string instructions =
518                         "SHIP_PLACEMENT:\n"
519                         "Разместите корабли: place размер,x,y,ориентация(H/V)\n"
520                         "Корабли: 1x4, 2x3, 3x2, 4x1\n"
521                         "Пример: place 4,0,0,H\n"
522                         "Когда готовы: ready";
523
524                     send_response_to(m.from, instructions.c_str());
525                     std::cout << m.from << " invited " << target
526                               << " to private game (ID: " << game_id << "). Creator
                                      auto-joined.\n";
527                 }
528             }
529         }
530         break;
531     }
532
533     case MSG_INVITE_TO_GAME: {
534         std::cout << "\n=== DEBUG: Processing MSG_INVITE_TO_GAME ===" << std::
                endl;
535
536         const char* target = m.payload;
537         ClientSlot* tgt = find_client(target);
538         ClientSlot* sender = find_client(m.from);
539
540         if (!tgt) {
541             std::cout << " Target '" << target << "' not found!" << std::endl;
542             send_response_to(m.from, "INVITE_FAIL:Игрок ненайден");
543             break;
544         }
545
546         if (!sender || sender->current_game_id == -1) {
547             std::cout << " Sender not in game" << std::endl;
548             send_response_to(m.from, "INVITE_FAIL:Вы невигре");
549             break;
550         }
551
552         Game* game = get_game(sender->current_game_id);
553         if (!game) {
554             std::cout << " Game not found" << std::endl;
555             send_response_to(m.from, "INVITE_FAIL:Игра ненайдена");
556             break;
557         }
558
559         std::string game_name = game->get_game_name();
560
561         char buf[RESP_MAX];
562         std::snprintf(buf, RESP_MAX, "INVITE:%s:%s:%d", m.from, game_name.c_str
                (),
563                       sender->current_game_id);
```

```
564
565         std::cout << " Sending to '" << target << "': " << buf << std::endl;
566         send_response_to(target, buf);
567         send_response_to(m.from, "INVITE_SENT:Приглашение отправлено");
568
569         break;
570     }
571
572     case MSG_CREATE: {
573         std::string game_name = m.payload;
574
575         if (game_name.empty()) {
576             send_response_to(m.from, "CREATE_FAIL:Имя игрынеможетбытьпустым");
577             break;
578         }
579
580         if (game_name.length() > LOGIN_MAX - 1) {
581             send_response_to(m.from, "CREATE_FAIL:Имя игрыслишкомдлинное");
582             break;
583         }
584
585         ClientSlot* client = find_client(m.from);
586         if (!client) {
587             send_response_to(m.from, "CREATE_FAIL:Вы незарегистрированы");
588             break;
589         }
590
591         if (client->current_game_id != -1) {
592             Game* existing_game = get_game(client->current_game_id);
593             if (existing_game && existing_game->has_player(m.from)) {
594                 send_response_to(m.from, "CREATE_FAIL:Вы ужевигре");
595                 break;
596             } else {
597                 client->current_game_id = -1;
598                 client->setup_complete = false;
599             }
600         }
601
602         for (int i = 0; i < 16; i++) {
603             if (root->games[i].used &&
604                 std::strcmp(root->games[i].game_name, game_name.c_str()) == 0) {
605                 send_response_to(m.from, "CREATE_FAIL:Игра стакимименемужесуществ
                    ует");
606                 break;
607             }
608         }
609
610         int game_id = create_public_game(game_name, m.from);
611         if (game_id == -1) {
612             send_response_to(m.from, "CREATE_FAIL:Сервер переполнен");
613         } else if (game_id == -2) {
```

```
614             send_response_to(m.from, "CREATE_FAIL:Игра стакимименемужесуществует"
                    );
615         } else {
616             char buf[RESP_MAX];
617             std::snprintf(buf, RESP_MAX, "GAME_CREATED:Публичная игра'%s' создана
                    (ID: %d)",
618                     game_name.c_str(), game_id);
619             send_response_to(m.from, buf);
620         }
621         break;
622     }
623     case MSG_JOIN: {
624         std::string target = m.payload;
625 
626         ClientSlot* client = find_client(m.from);
627         if (!client) {
628             send_response_to(m.from, "JOIN_FAIL:Вы незарегистрированы");
629             break;
630         }
631 
632         if (client->current_game_id != -1) {
633             send_response_to(m.from, "JOIN_FAIL:Вы ужевигре");
634             break;
635         }
636 
637         Game* game = nullptr;
638         int game_id = -1;
639 
640         if (isdigit(target[0])) {
641             game_id = std::stoi(target);
642             game = get_game(game_id);
643         }
644 
645         if (!game) {
646             for (auto& pair : games_map) {
647                 if (pair.second->get_game_name() == target) {
648                     game = pair.second;
649                     game_id = pair.first;
650                     break;
651                 }
652             }
653         }
654 
655         if (!game) {
656             send_response_to(m.from, "JOIN_FAIL:Игра ненайдена");
657             break;
658         }
659 
660         if (game->get_player1() == m.from || game->get_player2() == m.from) {
661             send_response_to(m.from, "JOIN_FAIL:Вы ужевэтойигре");
662             break;
```

```
663         }
664
665         if (game->get_player1()[0] && game->get_player2()[0]) {
666             send_response_to(m.from, "JOIN_FAIL:Игра ужезаполнена");
667             break;
668         }
669
670         if (game->join(m.from)) {
671             client->current_game_id = game_id;
672
673             std::string creator = game->get_player1();
674             if (creator.empty())
675                 creator = game->get_player2();
676
677             send_response_to(m.from, "JOIN_OK:Вы присоединилиськигре");
678
679             if (!creator.empty() && creator != m.from) {
680                 send_response_to(creator.c_str(), "OPPONENT_JOINED:Игрок присоеди
                        нился");
681             }
682
683             std::string instructions = "SHIP_PLACEMENT:\n"
684                                        "Разместите корабли: place размер,x,y,ориент
                                            ация(H/V)\n"
685                                        "Корабли: 1x4, 2x3, 3x2, 4x1\n"
686                                        "Пример: place 4,0,0,H\n"
687                                        "Когда готовы: ready";
688
689             send_response_to(m.from, instructions.c_str());
690             send_response_to(creator.c_str(), instructions.c_str());
691         } else {
692             send_response_to(m.from, "JOIN_FAIL:Не удалосьприсоединиться");
693         }
694         break;
695     }
696     case MSG_ACCEPT: {
697         int game_id = -1;
698         if (sscanf(m.payload, "%d", &game_id) == 1) {
699             Game* game = get_game(game_id);
700             if (!game) {
701                 send_response_to(m.from, "ACCEPT_FAIL:Игра ненайдена");
702             } else if (game->join(m.from)) {
703                 ClientSlot* client = find_client(m.from);
704                 if (client) {
705                     client->current_game_id = game_id;
706                 }
707
708                 send_response_to(m.from, "ACCEPT_OK:Вы присоединились");
709                 send_response_to(game->get_player1().c_str(),
710                                  "OPPONENT_JOINED:Игрок принялприглашение");
711
```

```
712                 std::string instructions =
713                     "SHIP_PLACEMENT:\n"
714                     "Разместите корабликомандой: place размер,x,y,ориентация(H/V)"
                            ;
715
716                 send_response_to(m.from, instructions.c_str());
717                 send_response_to(game->get_player1().c_str(), instructions.c_str
                        ());
718             } else {
719                 send_response_to(m.from, "ACCEPT_FAIL:Не удалосьприсоединиться");
720             }
721         } else {
722             send_response_to(m.from, "ACCEPT_FAIL:Неверный формат. ИспользуйтеID
                    игры");
723         }
724         break;
725     }
726     case MSG_PLACE_SHIP: {
727         std::cout << "DEBUG: Received PLACE_SHIP from " << m.from << " payload:
                " << m.payload
728                   << std::endl;
729
730         ClientSlot* client = find_client(m.from);
731         if (!client) {
732             send_response_to(m.from, "ERROR:Not registered");
733             break;
734         }
735
736         if (client->current_game_id == -1) {
737             send_response_to(m.from, "ERROR:Not in a game");
738             break;
739         }
740
741         Game* game = get_game(client->current_game_id);
742         if (!game) {
743             send_response_to(m.from, "ERROR:Game not found");
744             break;
745         }
746
747         uint8_t size, x, y;
748         bool horizontal;
749
750         int s, x_pos, y_pos;
751         char orientation;
752
753         if (sscanf(m.payload, "%d,%d,%d,%c", &s, &x_pos, &y_pos, &orientation)
                != 4) {
754             send_response_to(m.from, "SHIP_ERROR:Invalid format. Use: size,x,y,
                    orientation(H/V)");
755             break;
756         }
```

```
757
758         if (s < 1 || s > 4) {
759             send_response_to(m.from, "SHIP_ERROR:Size must be 1-4");
760             break;
761         }
762
763         if (x_pos < 0 || x_pos >= BOARD_SIZE || y_pos < 0 || y_pos >= BOARD_SIZE
                ) {
764             send_response_to(m.from, "SHIP_ERROR:Coordinates out of bounds");
765             break;
766         }
767
768         if (orientation != 'H' && orientation != 'V') {
769             send_response_to(m.from, "SHIP_ERROR:Orientation must be H or V");
770             break;
771         }
772
773         size = static_cast<uint8_t>(s);
774         x = static_cast<uint8_t>(x_pos);
775         y = static_cast<uint8_t>(y_pos);
776         horizontal = (orientation == 'H');
777
778         if (game->place_ship(m.from, size, x, y, horizontal)) {
779             send_response_to(m.from, "SHIP_PLACED:OK");
780             std::cout << "DEBUG: Ship placed successfully" << std::endl;
781         } else {
782             send_response_to(m.from, "SHIP_ERROR:Cannot place ship here");
783             std::cout << "DEBUG: Failed to place ship" << std::endl;
784         }
785         break;
786     }
787     case MSG_SETUP_COMPLETE: {
788         handle_setup_complete(m);
789         break;
790     }
791     case MSG_SHOT: {
792         ClientSlot* client = find_client(m.from);
793         if (!client || client->current_game_id == -1) {
794             send_response_to(m.from, "ERROR:Not in a game");
795             break;
796         }
797
798         Game* game = get_game(client->current_game_id);
799         if (!game) {
800             send_response_to(m.from, "ERROR:Game not found");
801             break;
802         }
803
804         if (!game->is_game_active()) {
805             send_response_to(m.from,
806                              "ERROR:Game not active. Wait for both players to
```

```
                                complete setup.");
807             break;
808         }
809
810         uint8_t x, y;
811         if (!parse_shot(m.payload, x, y)) {
812             send_response_to(m.from, "SHOT_FAIL:Invalid format. Use: x,y");
813             break;
814         }
815
816         if (!game->is_player_turn(m.from)) {
817             send_response_to(m.from, "ERROR:Not your turn");
818             break;
819         }
820
821         bool hit = game->make_shot(m.from, x, y);
822         std::string shooter = m.from;
823         std::string opponent =
824             (game->get_player1() == shooter) ? game->get_player2() : game->
                get_player1();
825
826         char buf[RESP_MAX];
827         if (hit) {
828             std::snprintf(buf, RESP_MAX, "SHOT_RESULT:HIT at %d,%d", x, y);
829         } else {
830             std::snprintf(buf, RESP_MAX, "SHOT_RESULT:MISS at %d,%d", x, y);
831         }
832         send_response_to(m.from, buf);
833
834         std::snprintf(buf, RESP_MAX, "OPPONENT_SHOT:%s at %d,%d:%s", shooter.
             c_str(), x, y,
835                       hit ? "HIT" : "MISS");
836         send_response_to(opponent.c_str(), buf);
837
838         std::string opponent_view = game->get_opponent_view(shooter);
839         send_response_to(m.from, ("OPPONENT_VIEW_UPDATE:\n" + opponent_view).
             c_str());
840
841         if (game->is_game_finished()) {
842             std::string winner = game->get_winner();
843             std::string loser =
844                 (winner == game->get_player1()) ? game->get_player2() : game->
                    get_player1();
845
846             send_response_to(winner.c_str(), " VICTORY:You won the game! ");
847             send_response_to(loser.c_str(), " DEFEAT:You lost the game ");
848
849             std::string winner_stats = game->get_statistics(winner);
850             std::string loser_stats = game->get_statistics(loser);
851
852             send_response_to(winner.c_str(), ("FINAL_STATS:\n" + winner_stats).
```

```
                c_str());
853             send_response_to(loser.c_str(), ("FINAL_STATS:\n" + loser_stats).
                    c_str());
854
855             remove_game(client->current_game_id);
856         } else {
857             std::string current_turn = game->get_current_turn();
858             if (current_turn == shooter && hit) {
859                 send_response_to(shooter.c_str(), "YOUR_TURN_AGAIN:You hit! Shoot
                        again");
860             } else if (current_turn == opponent) {
861                 send_response_to(opponent.c_str(), "YOUR_TURN:Make your move");
862             }
863         }
864         break;
865     }
866     case MSG_GET_BOARD: {
867         handle_get_board(m);
868         break;
869     }
870     case MSG_GET_OPPONENT_BOARD: {
871         handle_get_opponent_board(m);
872         break;
873     }
874     case MSG_GAME_STATUS: {
875         handle_game_status(m);
876         break;
877     }
878     case MSG_SURRENDER: {
879         handle_surrender(m);
880         break;
881     }
882     case MSG_LEAVE_GAME: {
883         ClientSlot* client = find_client(m.from);
884         if (client && client->current_game_id != -1) {
885             Game* game = get_game(client->current_game_id);
886             if (game) {
887                 // Проверяем, являетсялиигрокучастникомэтойигры
888                 if (!game->is_player_in_game(m.from)) {
889                     send_response_to(m.from, "ERROR:You are not in this game");
890                     break;
891                 }
892
893                 if (game->is_game_finished()) {
894                     send_response_to(m.from, "ERROR:Game already finished");
895                     break;
896                 }
897
898                 std::string other_player;
899                 if (game->get_player1() == m.from) {
900                     other_player = game->get_player2();
```

```
901                 } else {
902                     other_player = game->get_player1();
903                 }
904 
905                 game->remove_player(m.from);
906 
907                 if (!other_player.empty()) {
908                     std::string message =
909                         "OPPONENT_LEFT:Игрок " + std::string(m.from) + " вышелизиг
                            ры";
910                     send_response_to(other_player.c_str(), message.c_str());
911 
912                     if (game->is_waiting()) {
913                         send_response_to(other_player.c_str(),
914                                          "GAME_WAITING:Игра ожидаетновогоигрока");
915                     }
916                 }
917 
918                 client->current_game_id = -1;
919                 client->setup_complete = false;
920                 send_response_to(m.from, "LEFT_GAME:Вы вышлиизигры");
921 
922                 if (!game->get_player1()[0] && !game->get_player2()[0]) {
923                     remove_game(client->current_game_id);
924                 } else {
925                     if (!other_player.empty()) {
926                         ClientSlot* other_client = find_client(other_player.c_str
                                ());
927                         if (other_client) {
928                             other_client->setup_complete = false;
929                         }
930                     }
931                 }
932             }
933         } else {
934             send_response_to(m.from, "ERROR:You are not in any game");
935         }
936         break;
937     }
938     case MSG_QUIT: {
939         ClientSlot* c = find_client(m.from);
940         if (c) {
941             if (c->current_game_id != -1) {
942                 Game* game = get_game(c->current_game_id);
943                 if (game && !game->is_game_finished()) {
944                     std::string opponent =
945                         (game->get_player1() == m.from) ? game->get_player2() :
                            game->get_player1();
946                     send_response_to(opponent.c_str(), "OPPONENT_DISCONNECTED:You
                        win by forfeit");
947                     remove_game(c->current_game_id);
```

```
948                 }
949             }
950 
951             c->used = false;
952             c->has_response = false;
953             c->current_game_id = -1;
954             c->setup_complete = false;
955             std::memset(c->login, 0, LOGIN_MAX);
956             std::memset(c->response, 0, RESP_MAX);
957             std::cout << "Client quit: " << m.from << '\n';
958         }
959         break;
960     }
961     default:
962         send_response_to(m.from, "UNKNOWN_CMD");
963     }
964 }
965 
966 void Server::run() {
967     std::cout << "=== SERVER RUNNING ===\n";
968     while (true) {
969         pthread_mutex_lock(&root->mutex);
970         while (root->q_head == root->q_tail) {
971             pthread_cond_wait(&root->server_cond, &root->mutex);
972         }
973 
974         Message m = root->queue[root->q_head];
975         root->queue[root->q_head].used = false;
976         root->q_head = (root->q_head + 1) % QUEUE_SIZE;
977         pthread_mutex_unlock(&root->mutex);
978 
979         if (m.used) {
980             handle_message(m);
981         }
982     }
983 }
```

Листинг 5: Server.cpp - реализация сервера игры

```
1  #pragma once
2  #include "../include/SharedTypes.hpp"
3  #include "../include/SharedMemory.hpp"
4  #include <string>
5  #include <random>
6 
7  class Client {
8  public:
9      Client();
10     ~Client();
11 
```

```
12      void run();
13
14  private:
15      SharedMemory shm;
16      SharedMemoryRoot* root;
17      std::string login;
18      int current_game_id;
19      bool in_game;
20      bool in_setup;
21
22      bool setup_show_menu;
23
24      std::mt19937 rng;
25
26      std::string pending_invite_game_name;
27      std::string pending_invite_from;
28      int pending_invite_id;
29
30      bool enqueue_message(const Message& m);
31      bool wait_for_response(std::string &out, int timeout_ms = 1000);
32      ClientSlot* my_slot();
33      bool check_for_async_messages();
34      void handle_game_response(const std::string& response);
35
36      void show_main_menu();
37      void show_game_menu();
38      void place_ships_interactive();
39      void show_game_status();
40      void clear_response_buffer();
41
42      void auto_place_ships();
43      bool is_valid_position(uint8_t x, uint8_t y, uint8_t size, bool horizontal,
            const std::vector<std::pair<uint8_t, uint8_t>>& placed_positions);
44      void force_clear_response();
45      bool has_only_one_player() const;
46      bool is_player_in_game(const std::string& player) const;
47      void remove_player(const std::string& player);
48      void force_check_state();
49  };
```

Листинг 6: Client.hpp - интерфейс клиента игры

```
1   #include "Client.hpp"
2
3   #include <algorithm>
4   #include <chrono>
5   #include <cstring>
6   #include <iomanip>
7   #include <iostream>
8   #include <sstream>
```

```cpp
#include <vector>

Client::Client()
    : shm(false), root(shm.root()), current_game_id(-1), in_game(false), in_setup(false),
      pending_invite_id(-1), rng(std::random_device{}()) {
    if (!root)
        throw std::runtime_error("Cannot open shared memory; run server first");
}

void Client::force_check_state() {
    if (current_game_id != -1) {
        std::cout << " Проверяемсостояниеигры...\n";

        Message m;
        std::memset(&m, 0, sizeof(m));
        std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
        m.type = MSG_GAME_STATUS;

        clear_response_buffer();

        if (enqueue_message(m)) {
            std::string resp;
            if (wait_for_response(resp, 2000)) {
                if (resp.find("ERROR") != std::string::npos ||
                    resp.find("GAME_REMOVED") != std::string::npos ||
                    resp.find("Not in a game") != std::string::npos) {
                    std::cout << " Играненайдена, сбрасываемсостояние\n";
                    in_game = false;
                    in_setup = false;
                    current_game_id = -1;

                    ClientSlot* slot = my_slot();
                    if (slot) {
                        slot->current_game_id = -1;
                        slot->setup_complete = false;
                    }
                } else {
                    std::cout << " Играсуществует: " << resp.substr(0, 50) << "...\n";
                }
            } else {
                std::cout << " Нетответаотсервера, сбрасываемсостояние\n";
                in_game = false;
                in_setup = false;
                current_game_id = -1;
            }
        }
    }
}

```

```
58 bool Client::is_valid_position(uint8_t x, uint8_t y, uint8_t size, bool
       horizontal, const std::vector<std::pair<uint8_t, uint8_t>>& placed_positions
       ) {
59
60     if (horizontal) {
61         if (x + size > BOARD_SIZE)
62             return false;
63     } else {
64         if (y + size > BOARD_SIZE)
65             return false;
66     }
67
68     for (int i = 0; i < size; i++) {
69         int cx = horizontal ? x + i : x;
70         int cy = horizontal ? y : y + i;
71
72         for (int dy = -1; dy <= 1; dy++) {
73             for (int dx = -1; dx <= 1; dx++) {
74                 int nx = cx + dx;
75                 int ny = cy + dy;
76
77                 if (nx >= 0 && nx < BOARD_SIZE && ny >= 0 && ny < BOARD_SIZE) {
78                     for (const auto& pos : placed_positions) {
79                         if (pos.first == nx && pos.second == ny) {
80                             return false;
81                         }
82                     }
83                 }
84             }
85         }
86     }
87
88     return true;
89 }
90
91 void Client::auto_place_ships() {
92     std::cout << "\n" << std::string(50, '=') << "\n";
93     std::cout << " АВТОМАТИЧЕСКАЯРАССТАНОВКАКОРАБЛЕЙ\n";
94     std::cout << std::string(50, '=') << "\n";
95
96     std::vector<std::string> ships = {"4,0,0,H", "3,0,5,H", "3,0,8,H", "2,8,8,H"
           , "2,5,5,V",
97                                       "2,8,4,H", "1,6,0,H", "1,6,2,H", "1,3,2,H", "
                                           1,9,0,H"};
98
99     int placed_ships = 0;
100    int total_ships = ships.size();
101
102    clear_response_buffer();
103
104    for (const auto& ship_cmd : ships) {
```

```
105         std::cout << "Размещаем корабль: " << ship_cmd << "... ";
106 
107         Message m;
108         std::memset(&m, 0, sizeof(m));
109         std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
110         m.type = MSG_PLACE_SHIP;
111         std::strncpy(m.payload, ship_cmd.c_str(), CMD_MAX - 1);
112 
113         if (!enqueue_message(m)) {
114             std::cout << " Очередьпереполнена\n";
115             continue;
116         }
117 
118         std::string resp;
119         if (wait_for_response(resp, 2000)) {
120             if (resp.find("SHIP_PLACED") != std::string::npos ||
121                 resp.find("OK") != std::string::npos ||
122                 resp.find("YOUR_BOARD") != std::string::npos) {
123                 placed_ships++;
124                 std::cout << " Успешно\n";
125             } else {
126                 std::cout << " Ошибка: " << resp.substr(0, 50) << "\n";
127             }
128         } else {
129             std::cout << " Нетответаотсервера\n";
130         }
131 
132         usleep(100 * 1000);
133     }
134 
135     std::cout << "\n" << std::string(50, '-') << "\n";
136     std::cout << " РАССТАНОВКАЗАВЕРШЕНА\n";
137     std::cout << " Размещенокораблей: " << placed_ships << "/" << total_ships <<
        "\n";
138 
139     if (placed_ships == total_ships) {
140         std::cout << "  Всекораблиуспешноразмещены!\n";
141 
142         std::cout << " Показываемполе...\n";
143         Message board_msg;
144         std::memset(&board_msg, 0, sizeof(board_msg));
145         std::strncpy(board_msg.from, login.c_str(), LOGIN_MAX - 1);
146         board_msg.type = MSG_GET_BOARD;
147 
148         clear_response_buffer();
149 
150         if (enqueue_message(board_msg)) {
151             std::string resp;
152             if (wait_for_response(resp, 2000)) {
153                 std::cout << "\n" << resp << "\n";
154             }
```

```
155         }
156
157         std::cout << " Длязавершениярасстановкивведите'ready'\n";
158     } else {
159         std::cout << "  Невсекораблиудалосьразместить\n";
160         std::cout << " Завершитерасстановкувручную\n";
161     }
162
163     std::cout << std::string(50, '=') << "\n\n";
164 }
165
166 Client::~Client() {
167 }
168
169 ClientSlot* Client::my_slot() {
170     for (size_t i = 0; i < MAX_CLIENTS; ++i) {
171         if (root->clients[i].used &&
172             std::strncmp(root->clients[i].login, login.c_str(), LOGIN_MAX) == 0)
                {
173             return &root->clients[i];
174         }
175     }
176     return nullptr;
177 }
178
179 bool Client::enqueue_message(const Message& m) {
180     pthread_mutex_lock(&root->mutex);
181
182     size_t next_tail = (root->q_tail + 1) % QUEUE_SIZE;
183     if (next_tail == root->q_head) {
184         pthread_mutex_unlock(&root->mutex);
185         return false;
186     }
187
188     root->queue[root->q_tail] = m;
189     root->queue[root->q_tail].used = true;
190     root->q_tail = next_tail;
191
192     pthread_cond_signal(&root->server_cond);
193     pthread_mutex_unlock(&root->mutex);
194     return true;
195 }
196
197 bool Client::wait_for_response(std::string& out, int timeout_ms) {
198     auto start = std::chrono::steady_clock::now();
199
200     while (std::chrono::duration_cast<std::chrono::milliseconds>(std::chrono::
        steady_clock::now() -
201                                                                  start)
202                .count() < timeout_ms) {
203
```

```
204         pthread_mutex_lock(&root->mutex);
205         ClientSlot* slot = my_slot();
206 
207         if (!slot) {
208             pthread_mutex_unlock(&root->mutex);
209             usleep(100 * 1000);
210             continue;
211         }
212 
213         if (slot->has_response) {
214             out = slot->response;
215             slot->has_response = false;
216             std::memset(slot->response, 0, RESP_MAX);
217             pthread_mutex_unlock(&root->mutex);
218             return true;
219         }
220 
221         pthread_mutex_unlock(&root->mutex);
222         usleep(100 * 1000);
223     }
224 
225     return false;
226 }
227 
228 bool Client::check_for_async_messages() {
229     std::string resp;
230     if (wait_for_response(resp, 20)) {
231         std::cout << "[DEBUG] check_for_async_messages got: "
232                   << (resp.length() > 50 ? resp.substr(0, 50) + "..." : resp) <<
                          std::endl;
233         handle_game_response(resp);
234         return true;
235     }
236     return false;
237 }
238 
239 void Client::handle_game_response(const std::string& response) {
240     if (response.find("GAME_REMOVED:") == 0) {
241         std::cout << "\n " << response.substr(13) << "\n";
242         in_game = false;
243         in_setup = false;
244         current_game_id = -1;
245         pending_invite_game_name.clear();
246         pending_invite_from.clear();
247         pending_invite_id = -1;
248 
249         ClientSlot* slot = my_slot();
250         if (slot) {
251             slot->current_game_id = -1;
252             slot->setup_complete = false;
253         }
```

```
254     } else if (response.find("GAME_CREATED:") == 0) {
255         std::cout << "\n " << response.substr(13) << "\n";
256         in_game = true;
257         in_setup = true;
258 
259         size_t id_pos = response.find("ID:");
260         if (id_pos != std::string::npos) {
261             std::string id_str = response.substr(id_pos + 3);
262             id_str.erase(std::remove_if(id_str.begin(), id_str.end(),
263                                         [](char c) { return !std::isdigit(c); }),
264                          id_str.end());
265             if (!id_str.empty()) {
266                 current_game_id = std::stoi(id_str);
267             }
268         }
269     }
270 
271     if (response.find("INVITE:") == 0) {
272         std::cout << " DEBUG: Processing invitation: " << response << std::endl;
273 
274         size_t first_colon = response.find(':');
275         size_t second_colon = response.find(':', first_colon + 1);
276         size_t third_colon = response.find(':', second_colon + 1);
277 
278         if (first_colon != std::string::npos && second_colon != std::string::
                npos &&
279             third_colon != std::string::npos) {
280 
281             std::string inviter = response.substr(first_colon + 1, second_colon -
                     first_colon - 1);
282             std::string game_name =
283                 response.substr(second_colon + 1, third_colon - second_colon - 1)
                        ;
284             std::string game_id_str = response.substr(third_colon + 1);
285 
286             game_id_str.erase(std::remove_if(game_id_str.begin(), game_id_str.end
                    (),
287                                              [](char c) { return !std::isdigit(c);
                                                    }),
288                               game_id_str.end());
289 
290             std::cout << "\n" << std::string(50, '=') << "\n";
291             std::cout << "  ПРИГЛАШЕНИЕВИГРУ!\n";
292             std::cout << std::string(50, '=') << "\n";
293             std::cout << " Игра: " << game_name << "\n";
294             std::cout << " Приглашает: " << inviter << "\n";
295             std::cout << " ID: " << game_id_str << "\n\n";
296             std::cout << " Принять: join " << game_id_str << "\n";
297             std::cout << " Отклонить: ignore\n";
298             std::cout << std::string(50, '=') << "\n";
299 
```

```
300             pending_invite_from = inviter;
301             pending_invite_game_name = game_name;
302 
303             try {
304                 pending_invite_id = std::stoi(game_id_str);
305             } catch (...) {
306                 pending_invite_id = -1;
307             }
308 
309             show_main_menu();
310         }
311     } else if (response.find("OPPONENT_JOINED:") == 0) {
312         std::cout << "\n Противникприсоединился! Начинайтерасставлятькорабли.\n"
                ;
313     } else if (response.find("YOUR_BOARD:") == 0) {
314         std::cout << "\n" << response.substr(11) << "\n";
315     } else if (response.find("OPPONENT_VIEW:") == 0) {
316         std::cout << "\n" << response.substr(14) << "\n";
317     } else if (response.find("YOUR_TURN:") == 0) {
318         std::cout << "\n ВАШХОД! " << response.substr(10) << "\n";
319     } else if (response.find("YOUR_TURN_AGAIN:") == 0) {
320         std::cout << "\n ВАШХОДСНОВА! " << response.substr(16) << "\n";
321     } else if (response.find("OPPONENT_SHOT:") == 0) {
322         std::cout << "\n ПРОТИВНИКСТРЕЛЯЕТ: " << response.substr(14) << "\n";
323     } else if (response.find("SHOT_RESULT:") == 0) {
324         std::cout << "\n РЕЗУЛЬТАТВЫСТРЕЛА: " << response.substr(12) << "\n";
325     } else if (response.find("VICTORY:") == 0) {
326         std::cout << "\n" << std::string(50, '=') << "\n";
327         std::cout << "  ПОБЕДА! \n";
328         std::cout << " " << response.substr(8) << "\n";
329         std::cout << std::string(50, '=') << "\n\n";
330         in_game = false;
331         in_setup = false;
332         current_game_id = -1;
333     } else if (response.find("DEFEAT:") == 0) {
334         std::cout << "\n" << std::string(50, '=') << "\n";
335         std::cout << "  ПОРАЖЕНИЕ\n";
336         std::cout << " " << response.substr(7) << "\n";
337         std::cout << std::string(50, '=') << "\n\n";
338         in_game = false;
339         in_setup = false;
340         current_game_id = -1;
341     } else if (response.find("ACCEPT_OK") == 0 || response.find("JOIN_OK") == 0)
            {
342         std::cout << "\n Выприсоединилиськигре!\n";
343         in_game = true;
344         in_setup = true;
345         pending_invite_game_name.clear();
346         pending_invite_from.clear();
347         pending_invite_id = -1;
348     } else if (response.find("SHIP_PLACEMENT:") == 0) {
```

```
349         std::cout << "\n" << response << "\n";
350         if (!in_game) {
351             in_game = true;
352             in_setup = true;
353         }
354     } else if (response.find("LEFT_GAME:") == 0) {
355         std::cout << "\n " << response.substr(10) << "\n";
356         in_game = false;
357         in_setup = false;
358         current_game_id = -1;
359     } else if (response.find("GAME_CREATED") == 0) {
360         std::cout << "\n " << response.substr(13) << "\n";
361         in_game = true;
362         in_setup = true;
363 
364         size_t id_pos = response.find("ID:");
365         if (id_pos != std::string::npos) {
366             std::string id_str = response.substr(id_pos + 3);
367             id_str.erase(std::remove_if(id_str.begin(), id_str.end(),
368                                         [](char c) { return !std::isdigit(c); }),
369                          id_str.end());
370             if (!id_str.empty()) {
371                 current_game_id = std::stoi(id_str);
372             }
373         }
374     }
375     if (response.find("INVITE_SENT") == 0) {
376         std::cout << "\n " << response.substr(12) << "\n";
377         in_game = true;
378         in_setup = true;
379         std::cout << " Вывошливигру. Начинайтерасстановкукораблей!\n";
380     } else if (response.find("SETUP_COMPLETE") == 0) {
381         std::cout << "\n " << response.substr(15) << "\n";
382         in_setup = false;
383     } else if (response.find("OPPONENT_VIEW_UPDATE:") == 0) {
384         std::cout << "\n" << response.substr(21) << "\n";
385     } else if (response.find("GAME_STATUS:") == 0) {
386         std::cout << "\n" << response.substr(12) << "\n";
387     } else if (response.find("FINAL_STATS:") == 0) {
388         std::cout << "\n " << response.substr(12) << "\n";
389     } else if (response.find("ERROR:") == 0 || response.find("FAIL:") == 0 ||
390                response.find("INVALID") == 0 || response.find("SHIP_ERROR") == 0)
                       {
391         std::cout << "\n " << response << "\n";
392     } else if (response.find("REGISTERED:") == 0) {
393         std::cout << "\n " << response.substr(11) << "\n";
394     } else if (response.find("LEFT_GAME:") == 0) {
395         std::cout << "\n " << response.substr(10) << "\n";
396         in_game = false;
397         in_setup = false;
398         current_game_id = -1;
```

```
399     } else if (!response.empty() && response.find("===") != 0) {
400         if (response != "\n" && response.length() > 2) {
401             std::cout << "\n" << response << "\n";
402         }
403     }
404 }
405 void Client::show_main_menu() {
406     std::cout << "\n" << std::string(50, '=') << "\n";
407     std::cout << " МОРСКОЙБОЙ\n";
408     std::cout << std::string(50, '=') << "\n";
409     std::cout << " 1 - Списокигроковиигр\n";
410     std::cout << " 2 - Создатьпубличнуюигру\n";
411     std::cout << " 3 - Присоединитьсякигре\n";
412     std::cout << " 4 - Пригласитьигрока\n";
413     std::cout << " 5 - Проверитьприглашения\n";
414     std::cout << " 6 - Выйти\n";
415 
416     if (pending_invite_id != -1) {
417         std::cout << std::string(50, '=') << "\n";
418         std::cout << "  АКТИВНОЕПРИГЛАШЕНИЕ:\n";
419         std::cout << " Игра: " << pending_invite_game_name << "\n";
420         std::cout << " От: " << pending_invite_from << "\n";
421         std::cout << " ID: " << pending_invite_id << "\n";
422         std::cout << " Принять: join " << pending_invite_id << "\n";
423         std::cout << std::string(50, '=') << "\n";
424     }
425 
426     std::cout << " Выберитедействие: ";
427 }
428 
429 void Client::place_ships_interactive() {
430     std::cout << "\n" << std::string(50, '=') << "\n";
431     std::cout << " РАССТАНОВКАКОРАБЛЕЙ\n";
432     std::cout << std::string(50, '=') << "\n";
433     std::cout << " Формат: размер,x,y,ориентация(H/V)\n";
434     std::cout << " Пример: 4,0,0,H\n\n";
435     std::cout << " Кораблидляразмещения:\n";
436     std::cout << " 1 авианосец(4 клеток)\n";
437     std::cout << " 2 линкора(3 клетки)\n";
438     std::cout << " 3 крейсера(2 клетки)\n";
439     std::cout << " 4 эсминца(1 клетки)\n";
440     std::cout << std::string(50, '-') << "\n";
441     std::cout << " Команды:\n";
442     std::cout << " auto - автоматическаярасстановка\n";
443     std::cout << " ready - готовкигре\n";
444     std::cout << " board - посмотретьполе\n";
445     std::cout << " invite <логин> - пригласитьигрокавэтуигру\n";
446     std::cout << " menu - выйтивменю\n";
447     std::cout << std::string(50, '-') << "\n";
448 }
449 
```

```
450 void Client::show_game_menu() {
451     std::cout << "\n" << std::string(40, '=') << "\n";
452     std::cout << " ИГРАВПРОЦЕССЕ\n";
453     std::cout << std::string(40, '=') << "\n";
454     std::cout << " 1 - Сделатьвыстрел\n";
455     std::cout << " 2 - Посмотретьсвоеполе\n";
456     std::cout << " 3 - Посмотретьполепротивника\n";
457     std::cout << " 4 - Статусигры\n";
458     std::cout << " 5 - Сдаться\n";
459     std::cout << " 6 - Выйтивменю\n";
460     std::cout << std::string(40, '-') << "\n";
461     std::cout << " Выберитедействие: ";
462 }
463 
464 void Client::clear_response_buffer() {
465     pthread_mutex_lock(&root->mutex);
466     ClientSlot* slot = my_slot();
467     if (slot && slot->has_response) {
468         std::string resp = slot->response;
469         std::cout << "[DEBUG] Buffer has: " << resp << std::endl;
470 
471         if (resp.find("INVITE:") == 0) {
472             std::cout << "[DEBUG] Keeping invitation in buffer" << std::endl;
473         } else {
474             std::cout << "[DEBUG] Clearing buffer" << std::endl;
475             slot->has_response = false;
476             std::memset(slot->response, 0, RESP_MAX);
477         }
478     }
479     pthread_mutex_unlock(&root->mutex);
480 }
481 
482 void Client::run() {
483     std::cout << std::string(50, '=') << "\n";
484     std::cout << " ДОБРОПОЖАЛОВАТЬВМОРСКОЙГОЙ!\n";
485     std::cout << std::string(50, '=') << "\n";
486     std::cout << " Введитевашлогин: ";
487     std::getline(std::cin, login);
488 
489     if (login.empty()) {
490         std::cerr << "\n Логиннеможетбытьпустым\n";
491         return;
492     }
493 
494     Message reg;
495     std::memset(&reg, 0, sizeof(reg));
496     std::strncpy(reg.from, login.c_str(), LOGIN_MAX - 1);
497     reg.type = MSG_REGISTER;
498 
499     std::cout << "\n Регистрация...\n";
500     if (!enqueue_message(reg)) {
```

```
501         std::cerr << " Неудалосьотправитьзапрос\n";
502         return;
503     }
504 
505     std::string resp;
506     if (wait_for_response(resp, 2000)) {
507         handle_game_response(resp);
508     }
509 
510     bool running = true;
511 
512     while (running) {
513         static int check_counter = 0;
514         check_counter++;
515         if (check_counter >= 10 && current_game_id != -1) {
516             force_check_state();
517             check_counter = 0;
518         }
519 
520         for (int i = 0; i < 3; i++) {
521             if (check_for_async_messages()) {
522                 break;
523             }
524             usleep(50 * 1000);
525         }
526 
527         if (!in_game) {
528             show_main_menu();
529             std::string line;
530             std::getline(std::cin, line);
531 
532             if (line.find("join ") == 0 && !pending_invite_game_name.empty()) {
533                 std::string game_id_str = line.substr(5);
534 
535                 Message m;
536                 std::memset(&m, 0, sizeof(m));
537                 std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
538                 m.type = MSG_JOIN;
539                 std::strncpy(m.payload, game_id_str.c_str(), CMD_MAX - 1);
540 
541                 if (!enqueue_message(m)) {
542                     std::cout << "\n Очередьпереполнена\n";
543                 } else {
544                     if (wait_for_response(resp, 2000)) {
545                         handle_game_response(resp);
546                     }
547                 }
548             } else if (line == "ignore" && !pending_invite_game_name.empty()) {
549                 std::cout << "\n Приглашениепроигнорировано\n";
550                 pending_invite_game_name.clear();
551                 pending_invite_from.clear();
```

```
552                 pending_invite_id = -1;
553             } else if (line == "1") {
554                 Message m;
555                 std::memset(&m, 0, sizeof(m));
556                 std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
557                 m.type = MSG_LIST;
558 
559                 if (!enqueue_message(m)) {
560                     std::cout << "\n Очередьпереполнена\n";
561                 } else {
562                     if (wait_for_response(resp, 2000)) {
563                         std::cout << resp << "\n";
564                     }
565                 }
566             } else if (line == "2") {
567                 std::cout << "\n Введитеимядляновойигры: ";
568                 std::string game_name;
569                 std::getline(std::cin, game_name);
570 
571                 if (game_name.empty()) {
572                     std::cout << "\n Имяигрынеможетбытьпустым\n";
573                     continue;
574                 }
575 
576                 Message m;
577                 std::memset(&m, 0, sizeof(m));
578                 std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
579                 m.type = MSG_CREATE;
580                 std::strncpy(m.payload, game_name.c_str(), CMD_MAX - 1);
581 
582                 if (!enqueue_message(m)) {
583                     std::cout << "\n Очередьпереполнена\n";
584                 } else {
585                     if (wait_for_response(resp, 2000)) {
586                         handle_game_response(resp);
587                     }
588                 }
589             } else if (line == "3") {
590                 std::cout << "\n ВведитеимяилиID игры: ";
591                 std::string game_target;
592                 std::getline(std::cin, game_target);
593 
594                 if (game_target.empty()) {
595                     std::cout << "\n Имя/ID неможетбытьпустым\n";
596                     continue;
597                 }
598 
599                 Message m;
600                 std::memset(&m, 0, sizeof(m));
601                 std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
602                 m.type = MSG_JOIN;
```

```
603                 std::strncpy(m.payload, game_target.c_str(), CMD_MAX - 1);
604
605                 if (!enqueue_message(m)) {
606                     std::cout << "\n Очередьпереполнена\n";
607                 } else {
608                     if (wait_for_response(resp, 2000)) {
609                         handle_game_response(resp);
610                     }
611                 }
612             } else if (line == "4") {
613                 std::cout << "\n Введителогинигрокадляприглашения: ";
614                 std::string target;
615                 std::getline(std::cin, target);
616
617                 std::string game_name = login + "_vs_" + target + "_private";
618
619                 Message create_msg;
620                 std::memset(&create_msg, 0, sizeof(create_msg));
621                 std::strncpy(create_msg.from, login.c_str(), LOGIN_MAX - 1);
622                 create_msg.type = MSG_CREATE;
623                 std::strncpy(create_msg.payload, game_name.c_str(), CMD_MAX - 1);
624
625                 if (!enqueue_message(create_msg)) {
626                     std::cout << "\n Очередьпереполнена\n";
627                     continue;
628                 }
629
630                 std::string resp;
631                 if (wait_for_response(resp, 2000)) {
632                     if (resp.find("GAME_CREATED") != std::string::npos) {
633                         Message invite_msg;
634                         std::memset(&invite_msg, 0, sizeof(invite_msg));
635                         std::strncpy(invite_msg.from, login.c_str(), LOGIN_MAX -
                                1);
636                         invite_msg.type = MSG_INVITE_TO_GAME;
637                         std::strncpy(invite_msg.payload, target.c_str(), CMD_MAX -
                                 1);
638
639                         if (!enqueue_message(invite_msg)) {
640                             std::cout << "\n Очередьпереполнена\n";
641                         } else {
642                             std::string invite_resp;
643                             if (wait_for_response(invite_resp, 2000)) {
644                                 handle_game_response(invite_resp);
645                                 if (in_game && in_setup) {
646                                     place_ships_interactive();
647                                 }
648                             }
649                         }
650                     } else {
651                         handle_game_response(resp);
```

```
652                     }
653                 }
654 
655             } else if (line == "5") {
656                 std::cout << "\n Проверяемприглашения...\n";
657 
658                 bool found_invitation = false;
659                 for (int i = 0; i < 3; i++) {
660                     if (check_for_async_messages()) {
661                         found_invitation = true;
662                     }
663                     usleep(100 * 1000);
664                 }
665 
666                 if (!found_invitation && pending_invite_id == -1) {
667                     std::cout << " Приглашенийнет\n";
668                 } else if (pending_invite_id != -1) {
669                     std::cout << " Естьактивноеприглашение(ID: " <<
                        pending_invite_id << ")\n";
670                     std::cout << " Принять: join " << pending_invite_id << "\n";
671                 }
672             } else if (line == "6") {
673                 force_check_state();
674 
675                 std::cout << "\n Выуверены? (да/нет): ";
676                 std::string confirm;
677                 std::getline(std::cin, confirm);
678 
679                 std::string confirm_lower = confirm;
680                 std::transform(confirm_lower.begin(), confirm_lower.end(),
                    confirm_lower.begin(), ::tolower);
681 
682                 if (confirm_lower == "да" || confirm_lower == "y" ||
                    confirm_lower == "yes" ||
683                     confirm_lower == "д") {
684                     Message m;
685                     std::memset(&m, 0, sizeof(m));
686                     std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
687                     m.type = MSG_QUIT;
688                     enqueue_message(m);
689                     running = false;
690                     std::cout << "\n Выход...\n";
691                 }
692             } else if (line.find("join ") == 0) {
693                 std::string game_id_str = line.substr(5);
694 
695                 Message m;
696                 std::memset(&m, 0, sizeof(m));
697                 std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
698                 m.type = MSG_JOIN;
699                 std::strncpy(m.payload, game_id_str.c_str(), CMD_MAX - 1);
```

```
700
701                 if (!enqueue_message(m)) {
702                     std::cout << "\n Очередьпереполнена\n";
703                 } else {
704                     if (wait_for_response(resp, 2000)) {
705                         handle_game_response(resp);
706                         pending_invite_game_name.clear();
707                         pending_invite_from.clear();
708                         pending_invite_id = -1;
709                     }
710                 }
711             } else if (line == "ignore") {
712                 std::cout << "\n Приглашениеотклонено\n";
713                 pending_invite_game_name.clear();
714                 pending_invite_from.clear();
715                 pending_invite_id = -1;
716             } else {
717                 std::cout << "\n Невернаякоманда\n";
718             }
719         } else {
720             if (in_setup) {
721                 if (!check_for_async_messages()) {
722                     place_ships_interactive();
723                 }
724
725                 std::cout << "\n Команда: ";
726                 std::string command;
727                 std::getline(std::cin, command);
728
729                 clear_response_buffer();
730
731                 std::string cmd_lower = command;
732                 std::transform(cmd_lower.begin(), cmd_lower.end(), cmd_lower.
                    begin(), ::tolower);
733
734                 if (cmd_lower == "ready" || cmd_lower == "готово") {
735                     clear_response_buffer();
736
737                     Message m;
738                     std::memset(&m, 0, sizeof(m));
739                     std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
740                     m.type = MSG_SETUP_COMPLETE;
741
742                     std::cout << " Отправляем'ready' насервер...\n";
743
744                     if (!enqueue_message(m)) {
745                         std::cout << "\n Очередьпереполнена\n";
746                     } else {
747                         std::string resp;
748                         if (wait_for_response(resp, 3000)) {
749                             std::cout << " Полученответотсервера\n";
```

```
                        handle_game_response(resp);
                    } else {
                        std::cout << " Нетответаотсервера\n";
                    }
                }
            } else if (cmd_lower == "auto") {
                std::cout << "\n Запускаемавтоматическуюрасстановку...\n";
                auto_place_ships();
            } else if (cmd_lower == "board") {
                Message m;
                std::memset(&m, 0, sizeof(m));
                std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
                m.type = MSG_GET_BOARD;

                if (!enqueue_message(m)) {
                    std::cout << "\n Очередьпереполнена\n";
                } else {
                    if (wait_for_response(resp, 2000)) {
                        handle_game_response(resp);
                    }
                }
            }

            else if (cmd_lower.find("invite ") == 0) {
                std::string target = command.substr(7);

                if (target.empty() || target == login) {
                    std::cout << "\n Неверныйлогин\n";
                    continue;
                }

                Message m;
                std::memset(&m, 0, sizeof(m));
                std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
                m.type = MSG_INVITE_TO_GAME;
                std::strncpy(m.payload, target.c_str(), CMD_MAX - 1);

                if (!enqueue_message(m)) {
                    std::cout << "\n Очередьпереполнена\n";
                } else {
                    std::string resp;
                    if (wait_for_response(resp, 2000)) {
                        handle_game_response(resp);
                    }
                }
            }

            else if (cmd_lower == "menu") {
                Message m;
                std::memset(&m, 0, sizeof(m));
                std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
```

```
801                     m.type = MSG_LEAVE_GAME;
802                     enqueue_message(m);
803 
804                     in_game = false;
805                     in_setup = false;
806                     current_game_id = -1;
807                     std::cout << "\n Вывышлиизигры\n";
808                 } else {
809                     Message m;
810                     std::memset(&m, 0, sizeof(m));
811                     std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
812                     m.type = MSG_PLACE_SHIP;
813                     std::strncpy(m.payload, command.c_str(), CMD_MAX - 1);
814 
815                     if (!enqueue_message(m)) {
816                         std::cout << "\n Очередьпереполнена\n";
817                     } else {
818                         if (wait_for_response(resp, 2000)) {
819                             handle_game_response(resp);
820                         }
821                     }
822                 }
823             } else {
824                 bool has_async = check_for_async_messages();
825 
826                 if (has_async) {
827                     std::cout << "\nНажмите Enter дляпродолжения...";
828                     std::string dummy;
829                     std::getline(std::cin, dummy);
830                 }
831 
832                 show_game_menu();
833 
834                 std::string line;
835                 std::getline(std::cin, line);
836 
837                 if (line == "1") {
838                     std::cout << "\n Координатывыстрела(x,y): ";
839                     std::string shot;
840                     std::getline(std::cin, shot);
841 
842                     clear_response_buffer();
843 
844                     Message m;
845                     std::memset(&m, 0, sizeof(m));
846                     std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
847                     m.type = MSG_SHOT;
848                     std::strncpy(m.payload, shot.c_str(), CMD_MAX - 1);
849 
850                     std::cout << " Отправляемвыстрел...\n";
851 
```

```
852                     if (!enqueue_message(m)) {
853                         std::cout << "\n Очередьпереполнена\n";
854                     } else {
855                         std::string resp;
856                         if (wait_for_response(resp, 3000)) {
857                             std::cout << " Ответсервераполучен\n";
858                             handle_game_response(resp);
859                         } else {
860                             std::cout << " Нетответаотсервера\n";
861                         }
862                     }
863                 } else if (line == "2") {
864                     Message m;
865                     std::memset(&m, 0, sizeof(m));
866                     std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
867                     m.type = MSG_GET_BOARD;
868 
869                     if (!enqueue_message(m)) {
870                         std::cout << "\n Очередьпереполнена\n";
871                     } else {
872                         if (wait_for_response(resp, 2000)) {
873                             handle_game_response(resp);
874                         }
875                     }
876                 } else if (line == "3") {
877                     Message m;
878                     std::memset(&m, 0, sizeof(m));
879                     std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
880                     m.type = MSG_GET_OPPONENT_BOARD;
881 
882                     if (!enqueue_message(m)) {
883                         std::cout << "\n Очередьпереполнена\n";
884                     } else {
885                         if (wait_for_response(resp, 2000)) {
886                             handle_game_response(resp);
887                         }
888                     }
889                 } else if (line == "4") {
890                     Message m;
891                     std::memset(&m, 0, sizeof(m));
892                     std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
893                     m.type = MSG_GAME_STATUS;
894 
895                     if (!enqueue_message(m)) {
896                         std::cout << "\n Очередьпереполнена\n";
897                     } else {
898                         if (wait_for_response(resp, 2000)) {
899                             handle_game_response(resp);
900                         }
901                     }
902                 } else if (line == "5") {
```

```
903
904                     force_check_state();
905
906                     std::cout << "\n Выуверены? (да/нет): ";
907                     std::string confirm;
908                     std::getline(std::cin, confirm);
909
910                     std::string confirm_lower = confirm;
911                     std::transform(confirm_lower.begin(), confirm_lower.end(),
912                                    confirm_lower.begin(), ::tolower);
913
914                     if (confirm_lower == "да" || confirm_lower == "y" ||
                            confirm_lower == "yes" ||
915                         confirm_lower == "д") {
916                         Message m;
917                         std::memset(&m, 0, sizeof(m));
918                         std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
919                         m.type = MSG_SURRENDER;
920
921                         if (!enqueue_message(m)) {
922                             std::cout << "\n Очередьпереполнена\n";
923                         } else {
924                             if (wait_for_response(resp, 2000)) {
925                                 handle_game_response(resp);
926                             }
927                         }
928                     }
929                 } else if (line == "6") {
930                     std::cout << "\n Выходприравниваетсяксдаче!\n";
931                     std::cout << "Вы уверены? (да/нет): ";
932                     std::string confirm;
933                     std::getline(std::cin, confirm);
934
935                     std::string confirm_lower = confirm;
936                     std::transform(confirm_lower.begin(), confirm_lower.end(),
937                                    confirm_lower.begin(), ::tolower);
938
939                     if (confirm_lower == "да" || confirm_lower == "y" ||
                            confirm_lower == "yes" ||
940                         confirm_lower == "д") {
941                         Message m;
942                         std::memset(&m, 0, sizeof(m));
943                         std::strncpy(m.from, login.c_str(), LOGIN_MAX - 1);
944                         m.type = MSG_LEAVE_GAME;
945
946                         clear_response_buffer();
947
948                         if (!enqueue_message(m)) {
949                             std::cout << "\n Очередьпереполнена\n";
950                         } else {
951                             std::string resp;
```

```
                        if (wait_for_response(resp, 3000)) {
                            handle_game_response(resp);
                        } else {
                            std::cout
                                << " Нетответаотсервера, сбрасываемсостояниелок
                                    ально\n";
                            in_game = false;
                            in_setup = false;
                            current_game_id = -1;
                        }
                    }
                }
            }
        }
    }

    std::cout << "\n" << std::string(50, '=') << "\n";
    std::cout << " ИГРАЗАВЕРШЕНА\n";
    std::cout << " Спасибозаигру!\n";
    std::cout << std::string(50, '=') << "\n";
}
```

Листинг 7: Client.cpp - реализация клиента игры

```
#pragma once
#include <string>
#include <vector>

#include "../include/SharedTypes.hpp"

class Game {
  public:
    Game(const std::string& name, const std::string& creator, SharedMemoryRoot*
        root,
         bool is_public = false);
    ~Game();

    bool join(const std::string& player2);
    bool place_ship(const std::string& player, uint8_t size, uint8_t x, uint8_t
        y, bool horizontal);
    bool make_shot(const std::string& shooter, uint8_t x, uint8_t y);
    bool is_setup_complete(const std::string& player) const;
    void set_setup_complete(const std::string& player);
    bool is_game_active() const;
    bool is_game_finished() const;
    std::string get_winner() const;
    std::string get_current_turn() const;
    std::string get_status() const;

```

```
24      std::string get_player_board(const std::string& player, bool show_ships =
            true) const;
25      std::string get_opponent_view(const std::string& player) const;
26      std::string get_statistics(const std::string& player) const;
27
28      int get_id() const;
29      std::string get_game_name() const {
30          return std::string(game_data->game_name);
31      }
32      std::string get_player1() const {
33          return std::string(game_data->player1);
34      }
35      std::string get_player2() const {
36          return std::string(game_data->player2);
37      }
38      bool is_public() const {
39          return game_data->is_public;
40      }
41      bool is_waiting() const {
42          return game_data->state == GAME_WAITING;
43      }
44
45      bool has_player(const std::string& player) const;
46      bool is_player_turn(const std::string& player) const;
47
48      bool has_only_one_player() const;
49      bool is_player_in_game(const std::string& player) const;
50      void remove_player(const std::string& player);
51      bool is_empty() const;
52      bool is_full() const;
53      int get_player_count() const;
54
55  private:
56      int game_id;
57      SharedMemoryRoot* root;
58      GameData* game_data;
59
60      bool can_place_ship(uint8_t size, uint8_t x, uint8_t y, bool horizontal,
61                          CellState board[BOARD_SIZE][BOARD_SIZE]) const;
62      void place_ship_on_board(uint8_t size, uint8_t x, uint8_t y, bool horizontal
            ,
63                               CellState board[BOARD_SIZE][BOARD_SIZE], Ship*
                                   ship_array,
64                               uint8_t& ship_count);
65      bool check_hit(uint8_t x, uint8_t y, CellState board[BOARD_SIZE][BOARD_SIZE
            ], Ship* ships,
66                     uint8_t ship_count, bool& sunk, uint8_t& sunk_ship_index);
67      bool check_game_over() const;
68      void switch_turn();
69
70      std::string board_to_string(CellState board[BOARD_SIZE][BOARD_SIZE], bool
```

```
        show_ships) const;
};
```

Листинг 8: Game.hpp - интерфейс игровой логики

```cpp
#include "Game.hpp"
#include <sstream>
#include <iomanip>
#include <algorithm>
#include <iostream>

Game::Game(const std::string& name, const std::string& creator, SharedMemoryRoot
    * root, bool is_public)
    : root(root) {

    game_id = -1;
    for (int i = 0; i < 16; i++) {
        if (!root->games[i].used) {
            game_id = i;
            game_data = &root->games[i];
            break;
        }
    }

    if (game_id == -1) {
        throw std::runtime_error("No free game slots");
    }

    std::memset(game_data, 0, sizeof(GameData));
    game_data->used = true;
    std::strncpy(game_data->game_name, name.c_str(), LOGIN_MAX - 1);
    std::strncpy(game_data->player1, creator.c_str(), LOGIN_MAX - 1);
    game_data->state = GAME_WAITING;
    game_data->is_public = is_public;
    game_data->ship_count1 = 0;
    game_data->ship_count2 = 0;
    game_data->hits1 = game_data->hits2 = 0;
    game_data->misses1 = game_data->misses2 = 0;
    game_data->sunk1 = game_data->sunk2 = 0;

    for (int i = 0; i < BOARD_SIZE; i++) {
        for (int j = 0; j < BOARD_SIZE; j++) {
            game_data->board1[i][j] = CELL_EMPTY;
            game_data->board2[i][j] = CELL_EMPTY;
        }
    }
}

bool Game::has_only_one_player() const {
    return (game_data->player1[0] != '\0' && game_data->player2[0] == '\0') ||
```

```
45          (game_data->player1[0] == '\0' && game_data->player2[0] != '\0');
46  }
47
48  bool Game::is_player_in_game(const std::string& player) const {
49      return player == std::string(game_data->player1) ||
50             player == std::string(game_data->player2);
51  }
52
53  void Game::remove_player(const std::string& player) {
54      if (player == std::string(game_data->player1)) {
55          game_data->player1[0] = '\0';
56          game_data->ship_count1 = 0;
57          game_data->hits1 = game_data->misses1 = game_data->sunk1 = 0;
58
59          for (int i = 0; i < BOARD_SIZE; i++) {
60              for (int j = 0; j < BOARD_SIZE; j++) {
61                  game_data->board1[i][j] = CELL_EMPTY;
62              }
63          }
64
65          for (int i = 0; i < 10; i++) {
66              game_data->ships1[i] = Ship();
67          }
68
69          if (game_data->state == GAME_ACTIVE || game_data->state == GAME_SETUP) {
70              game_data->state = GAME_WAITING;
71          }
72      }
73      else if (player == std::string(game_data->player2)) {
74          game_data->player2[0] = '\0';
75          game_data->ship_count2 = 0;
76          game_data->hits2 = game_data->misses2 = game_data->sunk2 = 0;
77
78          for (int i = 0; i < BOARD_SIZE; i++) {
79              for (int j = 0; j < BOARD_SIZE; j++) {
80                  game_data->board2[i][j] = CELL_EMPTY;
81              }
82          }
83
84          for (int i = 0; i < 10; i++) {
85              game_data->ships2[i] = Ship();
86          }
87
88          if (game_data->state == GAME_ACTIVE || game_data->state == GAME_SETUP) {
89              game_data->state = GAME_WAITING;
90          }
91      }
92
93      if (has_only_one_player()) {
94          game_data->current_turn[0] = '\0';
95      }
```

```
96
97      if (game_data->player1[0] == '\0' && game_data->player2[0] == '\0') {
98          game_data->used = false;
99      }
100 }
101
102 bool Game::is_empty() const {
103     return game_data->player1[0] == '\0' && game_data->player2[0] == '\0';
104 }
105
106 bool Game::is_full() const {
107     return game_data->player1[0] != '\0' && game_data->player2[0] != '\0';
108 }
109
110 int Game::get_player_count() const {
111     int count = 0;
112     if (game_data->player1[0] != '\0') count++;
113     if (game_data->player2[0] != '\0') count++;
114     return count;
115 }
116
117 Game::~Game() {
118     if (game_data) {
119         game_data->used = false;
120     }
121 }
122
123 int Game::get_id() const {
124     return game_id;
125 }
126
127 bool Game::join(const std::string& player2) {
128     if (std::string(game_data->player1) == player2 ||
129         std::string(game_data->player2) == player2) {
130         return false;
131     }
132
133     if (game_data->player1[0] == '\0') {
134         std::strncpy(game_data->player1, player2.c_str(), LOGIN_MAX - 1);
135     } else if (game_data->player2[0] == '\0') {
136         std::strncpy(game_data->player2, player2.c_str(), LOGIN_MAX - 1);
137     } else {
138         return false; // Обаместазаняты
139     }
140
141     if (game_data->player1[0] != '\0' && game_data->player2[0] != '\0') {
142         game_data->state = GAME_SETUP;
143     }
144
145     return true;
146 }
```

```
147
148 bool Game::can_place_ship(uint8_t size, uint8_t x, uint8_t y, bool horizontal,
149                           CellState board[BOARD_SIZE][BOARD_SIZE]) const {
150
151     std::cout << "DEBUG: Checking if can place ship size " << (int)size
152               << " at " << (int)x << "," << (int)y
153               << " " << (horizontal ? "H" : "V") << std::endl;
154
155     // Проверяемграницы
156     if (x >= BOARD_SIZE || y >= BOARD_SIZE) {
157         std::cout << "DEBUG: Coordinates out of bounds" << std::endl;
158         return false;
159     }
160
161     if (size == 1) {
162         for (int dy = -1; dy <= 1; dy++) {
163             for (int dx = -1; dx <= 1; dx++) {
164                 int nx = x + dx;
165                 int ny = y + dy;
166                 if (nx >= 0 && nx < BOARD_SIZE && ny >= 0 && ny < BOARD_SIZE) {
167                     if (board[ny][nx] == CELL_SHIP) {
168                         std::cout << "DEBUG: Cell " << nx << "," << ny << "
                                already has ship" << std::endl;
169                         return false;
170                     }
171                 }
172             }
173         }
174         return true;
175     }
176
177     if (horizontal) {
178         if (x + size > BOARD_SIZE) {
179             std::cout << "DEBUG: Ship goes beyond right border" << std::endl;
180             return false;
181         }
182         for (int i = 0; i < size; i++) {
183             for (int dy = -1; dy <= 1; dy++) {
184                 for (int dx = -1; dx <= 1; dx++) {
185                     int nx = x + i + dx;
186                     int ny = y + dy;
187                     if (nx >= 0 && nx < BOARD_SIZE && ny >= 0 && ny < BOARD_SIZE)
                            {
188                         if (board[ny][nx] == CELL_SHIP) {
189                             std::cout << "DEBUG: Cell " << nx << "," << ny << "
                                    already has ship" << std::endl;
190                             return false;
191                         }
192                     }
193                 }
194             }
```

```
195         }
196     } else {
197         if (y + size > BOARD_SIZE) {
198             std::cout << "DEBUG: Ship goes beyond bottom border" << std::endl;
199             return false;
200         }
201         for (int i = 0; i < size; i++) {
202             for (int dy = -1; dy <= 1; dy++) {
203                 for (int dx = -1; dx <= 1; dx++) {
204                     int nx = x + dx;
205                     int ny = y + i + dy;
206                     if (nx >= 0 && nx < BOARD_SIZE && ny >= 0 && ny < BOARD_SIZE)
                            {
207                         if (board[ny][nx] == CELL_SHIP) {
208                             std::cout << "DEBUG: Cell " << nx << "," << ny << "
                                    already has ship" << std::endl;
209                             return false;
210                         }
211                     }
212                 }
213             }
214         }
215     }
216 
217     std::cout << "DEBUG: Ship can be placed here" << std::endl;
218     return true;
219 }
220 
221 void Game::place_ship_on_board(uint8_t size, uint8_t x, uint8_t y, bool
        horizontal,
222                                CellState board[BOARD_SIZE][BOARD_SIZE], Ship*
                                    ship_array, uint8_t& ship_count) {
223     Ship ship;
224     ship.size = size;
225     ship.health = size;
226     ship.horizontal = horizontal;
227     ship.start_x = x;
228     ship.start_y = y;
229     ship.sunk = false;
230 
231     if (size == 1) {
232         board[y][x] = CELL_SHIP;
233     } else if (horizontal) {
234         for (int i = 0; i < size; i++) {
235             board[y][x + i] = CELL_SHIP;
236         }
237     } else {
238         for (int i = 0; i < size; i++) {
239             board[y + i][x] = CELL_SHIP;
240         }
241     }
```

```
242
243     ship_array[ship_count] = ship;
244     ship_count++;
245 }
246
247 bool Game::place_ship(const std::string& player, uint8_t size, uint8_t x,
        uint8_t y, bool horizontal) {
248     if (game_data->state != GAME_WAITING && game_data->state != GAME_SETUP) {
249         std::cout << "DEBUG: Wrong game state: " << (int)game_data->state << std
                ::endl;
250         return false;
251     }
252
253     bool valid_size = false;
254     int required_count = 0;
255
256     switch(size) {
257         case 4: valid_size = true; required_count = 1; break;
258         case 3: valid_size = true; required_count = 2; break;
259         case 2: valid_size = true; required_count = 3; break;
260         case 1: valid_size = true; required_count = 4; break;
261         default:
262             std::cout << "DEBUG: Invalid ship size: " << (int)size << std::endl;
263             return false;
264     }
265
266     if (!valid_size) return false;
267
268     if (size == 1) {
269         horizontal = true;
270     }
271
272     uint8_t current_count = 0;
273     Ship* ships = nullptr;
274     uint8_t* ship_count = nullptr;
275     CellState (*board)[BOARD_SIZE] = nullptr;
276
277     if (player == std::string(game_data->player1)) {
278         ships = game_data->ships1;
279         ship_count = &game_data->ship_count1;
280         board = game_data->board1;
281         std::cout << "DEBUG: Player 1 placing ship" << std::endl;
282     } else if (player == std::string(game_data->player2)) {
283         ships = game_data->ships2;
284         ship_count = &game_data->ship_count2;
285         board = game_data->board2;
286         std::cout << "DEBUG: Player 2 placing ship" << std::endl;
287     } else {
288         std::cout << "DEBUG: Unknown player: " << player << std::endl;
289         return false;
290     }
```

```
291
292     for (int i = 0; i < *ship_count; i++) {
293         if (ships[i].size == size) current_count++;
294     }
295
296     if (current_count >= required_count) {
297         std::cout << "DEBUG: Too many ships of size " << (int)size
298                   << " (have " << current_count << ", need " << required_count <<
                          ")" << std::endl;
299         return false;
300     }
301
302     if (!can_place_ship(size, x, y, horizontal, board)) {
303         std::cout << "DEBUG: Cannot place ship at " << (int)x << "," << (int)y
304                   << " size " << (int)size << (horizontal ? "H" : "V") << std::
                          endl;
305         return false;
306     }
307
308     place_ship_on_board(size, x, y, horizontal, board, ships, *ship_count);
309
310     std::cout << "DEBUG: Ship placed successfully. Player " << player
311               << " now has " << (int)*ship_count << " ships" << std::endl;
312
313     if (game_data->state == GAME_WAITING) {
314         game_data->state = GAME_SETUP;
315         std::cout << "DEBUG: Game state changed to SETUP" << std::endl;
316     }
317
318     return true;
319 }
320
321 bool Game::check_hit(uint8_t x, uint8_t y, CellState board[BOARD_SIZE][
        BOARD_SIZE],
322                      Ship* ships, uint8_t ship_count, bool& sunk, uint8_t&
                             sunk_ship_index) {
323     if (board[y][x] == CELL_SHIP) {
324         for (int i = 0; i < ship_count; i++) {
325             Ship& ship = ships[i];
326             if (ship.sunk) continue;
327
328             if (ship.horizontal) {
329                 if (y == ship.start_y && x >= ship.start_x && x < ship.start_x +
                        ship.size) {
330                     ship.health--;
331                     std::cout << "DEBUG: Hit ship " << i << " at " << (int)x << ",
                            " << (int)y
332                               << ". Health now: " << (int)ship.health << std::endl;
333
334                     if (ship.health == 0) {
335                         ship.sunk = true;
```

```
336                         sunk = true;
337                         sunk_ship_index = i;
338                         std::cout << "DEBUG: Ship " << i << " SUNK! Marking cells
                                ..." << std::endl;
339 
340                         // Помечаемвсеклеткикорабляка kпотопленные
341                         for (int j = 0; j < ship.size; j++) {
342                             board[ship.start_y][ship.start_x + j] = CELL_SUNK;
343                         }
344                     } else {
345                         board[y][x] = CELL_HIT;
346                     }
347                     return true;
348                 }
349             } else {
350                 if (x == ship.start_x && y >= ship.start_y && y < ship.start_y +
                        ship.size) {
351                     ship.health--;
352                     std::cout << "DEBUG: Hit ship " << i << " at " << (int)x << ",
                            " << (int)y
353                               << ". Health now: " << (int)ship.health << std::endl;
354 
355                     if (ship.health == 0) {
356                         ship.sunk = true;
357                         sunk = true;
358                         sunk_ship_index = i;
359                         std::cout << "DEBUG: Ship " << i << " SUNK! Marking cells
                                ..." << std::endl;
360 
361                         for (int j = 0; j < ship.size; j++) {
362                             board[ship.start_y + j][ship.start_x] = CELL_SUNK;
363                         }
364                     } else {
365                         board[y][x] = CELL_HIT;
366                     }
367                     return true;
368                 }
369             }
370         }
371     } else if (board[y][x] == CELL_EMPTY) {
372         board[y][x] = CELL_MISS;
373         return false;
374     }
375 
376     return false;
377 }
378 
379 bool Game::make_shot(const std::string& shooter, uint8_t x, uint8_t y) {
380     if (game_data->state != GAME_ACTIVE) return false;
381     if (!is_player_turn(shooter)) return false;
382     if (x >= BOARD_SIZE || y >= BOARD_SIZE) return false;
```

```
383
384     bool is_player1 = (shooter == std::string(game_data->player1));
385     bool hit = false;
386     bool sunk = false;
387     uint8_t sunk_ship_index = 0;
388
389     CellState (*target_board)[BOARD_SIZE] = nullptr;
390     Ship* target_ships = nullptr;
391     uint8_t target_ship_count = 0;
392
393     if (is_player1) {
394         target_board = game_data->board2;
395         target_ships = game_data->ships2;
396         target_ship_count = game_data->ship_count2;
397     } else {
398         target_board = game_data->board1;
399         target_ships = game_data->ships1;
400         target_ship_count = game_data->ship_count1;
401     }
402
403     hit = check_hit(x, y, target_board, target_ships, target_ship_count, sunk,
        sunk_ship_index);
404
405     if (is_player1) {
406         if (hit) {
407             game_data->hits1++;
408             if (sunk) {
409                 game_data->sunk1++;
410                 std::cout << "DEBUG: Player 1 sunk a ship! Total sunk: " << (int)
                    game_data->sunk1 << std::endl;
411             }
412         } else {
413             game_data->misses1++;
414         }
415     } else {
416         if (hit) {
417             game_data->hits2++;
418             if (sunk) {
419                 game_data->sunk2++;
420                 std::cout << "DEBUG: Player 2 sunk a ship! Total sunk: " << (int)
                    game_data->sunk2 << std::endl;
421             }
422         } else {
423             game_data->misses2++;
424         }
425     }
426
427     if (check_game_over()) {
428         game_data->state = GAME_FINISHED;
429         game_data->end_time = time(nullptr);
430         std::strcpy(game_data->current_turn, "");
```

```
431     } else if (!hit) {
432         switch_turn();
433     } else {
434         std::cout << "DEBUG: Hit but not sunk, shooter gets another turn" << std
                ::endl;
435     }
436 
437     return hit;
438 }
439 
440 void Game::switch_turn() {
441     if (std::string(game_data->current_turn) == std::string(game_data->player1))
             {
442         std::strcpy(game_data->current_turn, game_data->player2);
443     } else {
444         std::strcpy(game_data->current_turn, game_data->player1);
445     }
446 }
447 
448 bool Game::check_game_over() const {
449     bool all_sunk2 = true;
450     for (int i = 0; i < game_data->ship_count2; i++) {
451         if (!game_data->ships2[i].sunk) {
452             all_sunk2 = false;
453             break;
454         }
455     }
456 
457     bool all_sunk1 = true;
458     for (int i = 0; i < game_data->ship_count1; i++) {
459         if (!game_data->ships1[i].sunk) {
460             all_sunk1 = false;
461             break;
462         }
463     }
464 
465     return all_sunk1 || all_sunk2;
466 }
467 
468 bool Game::is_setup_complete(const std::string& player) const {
469     if (player == std::string(game_data->player1)) {
470         return game_data->ship_count1 == 10;
471     } else if (player == std::string(game_data->player2)) {
472         return game_data->ship_count2 == 10;
473     }
474     return false;
475 }
476 
477 void Game::set_setup_complete(const std::string& player) {
478     for (size_t i = 0; i < MAX_CLIENTS; i++) {
479         if (root->clients[i].used && std::strcmp(root->clients[i].login, player.
```

```
            c_str()) == 0) {
480             root->clients[i].setup_complete = true;
481             break;
482         }
483     }
484 
485     bool player1_ready = false;
486     bool player2_ready = false;
487 
488     for (size_t i = 0; i < MAX_CLIENTS; i++) {
489         if (root->clients[i].used) {
490             if (std::strcmp(root->clients[i].login, game_data->player1) == 0) {
491                 player1_ready = root->clients[i].setup_complete;
492             } else if (std::strcmp(root->clients[i].login, game_data->player2) ==
                     0) {
493                 player2_ready = root->clients[i].setup_complete;
494             }
495         }
496     }
497 
498     if (player1_ready && player2_ready) {
499         game_data->state = GAME_ACTIVE;
500         game_data->start_time = time(nullptr);
501         std::strcpy(game_data->current_turn, game_data->player1);
502     }
503 }
504 
505 bool Game::is_game_active() const {
506     return game_data->state == GAME_ACTIVE;
507 }
508 
509 bool Game::is_game_finished() const {
510     return game_data->state == GAME_FINISHED;
511 }
512 
513 std::string Game::get_winner() const {
514     if (!is_game_finished()) return "";
515 
516     bool all_sunk1 = true;
517     for (int i = 0; i < game_data->ship_count1; i++) {
518         if (!game_data->ships1[i].sunk) {
519             all_sunk1 = false;
520             break;
521         }
522     }
523 
524     bool all_sunk2 = true;
525     for (int i = 0; i < game_data->ship_count2; i++) {
526         if (!game_data->ships2[i].sunk) {
527             all_sunk2 = false;
528             break;
```

```
529         }
530     }
531 
532     if (all_sunk1) return std::string(game_data->player2);
533     if (all_sunk2) return std::string(game_data->player1);
534     return "";
535 }
536 
537 std::string Game::get_current_turn() const {
538     return std::string(game_data->current_turn);
539 }
540 
541 std::string Game::board_to_string(CellState board[BOARD_SIZE][BOARD_SIZE], bool
        show_ships) const {
542     std::stringstream ss;
543 
544     ss << " ";
545     for (int i = 0; i < BOARD_SIZE; i++) {
546         ss << std::setw(2) << i << " ";
547     }
548     ss << "\n";
549 
550     for (int y = 0; y < BOARD_SIZE; y++) {
551         ss << std::setw(2) << y << " ";
552         for (int x = 0; x < BOARD_SIZE; x++) {
553             char symbol = '.';
554             switch(board[y][x]) {
555                 case CELL_EMPTY: symbol = '.'; break;
556                 case CELL_SHIP: symbol = show_ships ? 'S' : '.'; break;
557                 case CELL_HIT: symbol = 'X'; break;
558                 case CELL_MISS: symbol = 'O'; break;
559                 case CELL_SUNK: symbol = '#'; break;
560             }
561             ss << " " << symbol << " ";
562         }
563         ss << "\n";
564     }
565 
566     return ss.str();
567 }
568 
569 std::string Game::get_player_board(const std::string& player, bool show_ships)
        const {
570     if (player == std::string(game_data->player1)) {
571         return board_to_string(game_data->board1, show_ships);
572     } else if (player == std::string(game_data->player2)) {
573         return board_to_string(game_data->board2, show_ships);
574     }
575     return "";
576 }
577 
```

```
578 std::string Game::get_opponent_view(const std::string& player) const {
579     CellState temp_board[BOARD_SIZE][BOARD_SIZE];
580 
581     if (player == std::string(game_data->player1)) {
582         for (int y = 0; y < BOARD_SIZE; y++) {
583             for (int x = 0; x < BOARD_SIZE; x++) {
584                 if (game_data->board2[y][x] == CELL_SHIP) {
585                     temp_board[y][x] = CELL_EMPTY;
586                 } else {
587                     temp_board[y][x] = game_data->board2[y][x];
588                 }
589             }
590         }
591     } else {
592         for (int y = 0; y < BOARD_SIZE; y++) {
593             for (int x = 0; x < BOARD_SIZE; x++) {
594                 if (game_data->board1[y][x] == CELL_SHIP) {
595                     temp_board[y][x] = CELL_EMPTY;
596                 } else {
597                     temp_board[y][x] = game_data->board1[y][x];
598                 }
599             }
600         }
601     }
602 
603     return board_to_string(temp_board, false);
604 }
605 
606 std::string Game::get_statistics(const std::string& player) const {
607     std::stringstream ss;
608 
609     bool is_player1 = (player == std::string(game_data->player1));
610 
611     ss << "Статистика:\n";
612     ss << "Сбито кораблей: " << (is_player1 ? (int)game_data->sunk1 : (int)
        game_data->sunk2) << "\n";
613     ss << "Попаданий: " << (is_player1 ? (int)game_data->hits1 : (int)game_data
        ->hits2) << "\n";
614     ss << "Промахов: " << (is_player1 ? (int)game_data->misses1 : (int)game_data
        ->misses2) << "\n";
615 
616     if (is_game_finished()) {
617         ss << "Игра завершена!\n";
618         std::string winner = get_winner();
619         if (winner == player) {
620             ss << "Вы победили!\n";
621         } else {
622             ss << "Победил: " << winner << "\n";
623         }
624     } else if (is_game_active()) {
625         ss << "Текущий ход: " << get_current_turn() << "\n";
```

```
626         if (get_current_turn() == player) {
627             ss << "Ваш ход!\n";
628         } else {
629             ss << "Ход противника\n";
630         }
631     }
632 
633     return ss.str();
634 }
635 
636 std::string Game::get_status() const {
637     std::stringstream ss;
638 
639     ss << "Игра: " << game_data->game_name << " (ID: " << game_id << ")\n";
640     ss << "Игрок 1: " << game_data->player1 << "\n";
641     ss << "Игрок 2: " << (game_data->player2[0] ? game_data->player2 : "ожидает
        ...") << "\n";
642     ss << "Тип: " << (game_data->is_public ? "публичная" : "приватная") << "\n";
643     ss << "Статус: ";
644 
645     switch(game_data->state) {
646         case GAME_WAITING: ss << "Ожидание второгоигрока"; break;
647         case GAME_SETUP: ss << "Расстановка кораблей"; break;
648         case GAME_ACTIVE: ss << "Идет игра(ход: " << game_data->current_turn <<
            ")"; break;
649         case GAME_FINISHED:
650             ss << "Завершена. Победитель: " << get_winner();
651             break;
652     }
653 
654     return ss.str();
655 }
656 
657 bool Game::has_player(const std::string& player) const {
658     return (player == std::string(game_data->player1) ||
659             player == std::string(game_data->player2));
660 }
661 
662 bool Game::is_player_turn(const std::string& player) const {
663     return (game_data->state == GAME_ACTIVE &&
664             std::string(game_data->current_turn) == player);
665 }
```

Листинг 9: Game.cpp - реализация игровой логики

```
1 #include "Server.hpp"
2 #include <iostream>
3 
4 int main() {
5     try {
```

```
        Server s;
        s.run();
    } catch (const std::exception &ex) {
        std::cerr << "Server error: " << ex.what() << std::endl;
        return 1;
    }
    return 0;
}
```

Листинг 10: main.cpp (сервер) - точка входа серверного приложения

```
#include "Client.hpp"
#include <iostream>

int main() {
    try {
        Client c;
        c.run();
    } catch (const std::exception &ex) {
        std::cerr << "Client error: " << ex.what() << std::endl;
        return 1;
    }
    return 0;
}
```

Листинг 11: main.cpp (клиент) - точка входа клиентского приложения

Системные вызовы сервера

```
execve("./server/server",["./server/server"],0x7ffea73e20d0 /* 41 vars */)
= 0
brk(NULL)                                       = 0x5d3e668db000
mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x7b5f05cd1000
access("/etc/ld.so.preload",R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC) = 3
fstat(3,{st_mode=S_IFREG|0644,st_size=34331,...}) = 0
mmap(NULL,34331,PROT_READ,MAP_PRIVATE,3,0) = 0x7b5f05cc8000
close(3)                                        = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libstdc++.so.6",O_RDONLY|O_CLOEXEC)
= 3
read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"...,832)
= 832
fstat(3,{st_mode=S_IFREG|0644,st_size=2592224,...}) = 0
mmap(NULL,2609472,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7b5f05a00000
mmap(0x7b5f05a9d000,1343488,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3
= 0x7b5f05a9d000
mmap(0x7b5f05be5000,552960,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x1e5000)
= 0x7b5f05be5000
```

```
mmap(0x7b5f05c6c000,57344,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7b5f05c6c000
mmap(0x7b5f05c7a000,12608,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,-1
= 0x7b5f05c7a000
close(3)                                = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libgcc_s.so.1",O_RDONLY|O_CLOEXEC) =
3
read(3,"\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"...,832)
= 832
fstat(3,{st_mode=S_IFREG|0644,st_size=183024,...}) = 0
mmap(NULL,185256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7b5f05c9a000
mmap(0x7b5f05c9e000,147456,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7b5f05c9e000
mmap(0x7b5f05cc2000,16384,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x28000)
= 0x7b5f05cc2000
mmap(0x7b5f05cc6000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0
= 0x7b5f05cc6000
close(3)                                = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libc.so.6",O_RDONLY|O_CLOEXEC) = 3
read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"...,832)
= 832
pread64(3,"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"...,784,64)
= 784
fstat(3,{st_mode=S_IFREG|0755,st_size=2125328,...}) = 0
pread64(3,"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"...,784,64)
= 784
mmap(NULL,2170256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7b5f05600000
mmap(0x7b5f05628000,1605632,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3
= 0x7b5f05628000
mmap(0x7b5f057b0000,323584,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x1b0000)
= 0x7b5f057b0000
mmap(0x7b5f057ff000,24576,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7b5f057ff000
mmap(0x7b5f05805000,52624,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,-1
= 0x7b5f05805000
close(3)                                = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libm.so.6",O_RDONLY|O_CLOEXEC) = 3
read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"...,832)
= 832
fstat(3,{st_mode=S_IFREG|0644,st_size=952616,...}) = 0
mmap(NULL,950296,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7b5f05917000
mmap(0x7b5f05927000,520192,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7b5f05927000
mmap(0x7b5f059a6000,360448,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x8f000)
= 0x7b5f059a6000
mmap(0x7b5f059fe000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0
= 0x7b5f059fe000
close(3)                                = 0
```

```
mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x7b5f05c98000
mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x7b5f05c95000
arch_prctl(ARCH_SET_FS,0x7b5f05c95740) = 0
set_tid_address(0x7b5f05c95a10)         = 56741
set_robust_list(0x7b5f05c95a20,24)      = 0
rseq(0x7b5f05c96060,0x20,0,0x53053053) = 0
mprotect(0x7b5f057ff000,16384,PROT_READ) = 0
mprotect(0x7b5f059fe000,4096,PROT_READ) = 0
mprotect(0x7b5f05cc6000,4096,PROT_READ) = 0
mprotect(0x7b5f05c6c000,45056,PROT_READ) = 0
mprotect(0x5d3e51d77000,4096,PROT_READ) = 0
mprotect(0x7b5f05d09000,8192,PROT_READ) = 0
prlimit64(0,RLIMIT_STACK,NULL,{rlim_cur=8192*1024,rlim_max=RLIM64_INFINITY})
= 0
munmap(0x7b5f05cc8000,34331)            = 0
futex(0x7b5f05c7a7bc,FUTEX_WAKE_PRIVATE,2147483647) = 0
getrandom("\x57\x21\xa7\xd5\x56\xe3\x4a\xf3",8,GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5d3e668db000
brk(0x5d3e668fc000)                     = 0x5d3e668fc000
openat(AT_FDCWD,"/dev/shm/battleship_shm_v3",O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,0666
= 3
ftruncate(3,68720)                      = 0
mmap(NULL,68720,PROT_READ|PROT_WRITE,MAP_SHARED,3,0) = 0x7b5f05c84000
fstat(1,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x5),...}) = 0
write(1,"Server: shared memory initialize"...,34) = 34
write(1,"=== SERVER RUNNING ===\n",23) = 23
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"Registered client: dmitriy\n",27) = 27
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAIT,2,NULL) = -1 EAGAIN (Resource temporarily unavailable
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"Registered client: alice\n",25) = 25
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"Public game 'game1'created by d"...,47) = 47
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"\n",1)                        = 1
write(1,"=== DEBUG: Processing MSG_INVITE"...,45) = 45
write(1,"\360\237\223\244 Sending to 'alice': INVITE:"...,48) = 48
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAIT,2,NULL) = -1 EAGAIN (Resource temporarily unavailable
```

```
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,57) = 57
write(1,"DEBUG: Player 1 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship can be placed here\n",31) = 31
write(1,"DEBUG: Ship placed successfully."...,64) = 64
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,57) = 57
write(1,"DEBUG: Player 1 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship can be placed here\n",31) = 31
write(1,"DEBUG: Ship placed successfully."...,64) = 64
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,57) = 57
write(1,"DEBUG: Player 1 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship can be placed here\n",31) = 31
write(1,"DEBUG: Ship placed successfully."...,64) = 64
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,57) = 57
write(1,"DEBUG: Player 1 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship can be placed here\n",31) = 31
write(1,"DEBUG: Ship placed successfully."...,64) = 64
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,57) = 57
write(1,"DEBUG: Player 1 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship can be placed here\n",31) = 31
write(1,"DEBUG: Ship placed successfully."...,64) = 64
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
```

```
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,57) = 57
write(1,"DEBUG: Player 1 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship can be placed here\n",31) = 31
write(1,"DEBUG: Ship placed successfully."...,64) = 64
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAIT,2,NULL) = -1 EAGAIN (Resource temporarily unavailable
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,57) = 57
write(1,"DEBUG: Player 1 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship placed successfully."...,64) = 64
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,57) = 57
write(1,"DEBUG: Player 1 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship placed successfully."...,64) = 64
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAIT,2,NULL) = 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,57) = 57
write(1,"DEBUG: Player 1 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship placed successfully."...,64) = 64
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAIT,2,NULL) = -1 EAGAIN (Resource temporarily unavailable
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,57) = 57
write(1,"DEBUG: Player 1 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship placed successfully."...,65) = 65
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAIT,2,NULL) = -1 EAGAIN (Resource temporarily unavailable
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
```

```
futex(0x7b5f05c84000,FUTEX_WAIT,2,NULL) = -1 EAGAIN (Resource temporarily unavailable
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,55) = 55
write(1,"DEBUG: Player 2 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship can be placed here\n",31) = 31
write(1,"DEBUG: Ship placed successfully."...,62) = 62
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAIT,2,NULL) = -1 EAGAIN (Resource temporarily unavailable
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,55) = 55
write(1,"DEBUG: Player 2 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship can be placed here\n",31) = 31
write(1,"DEBUG: Ship placed successfully."...,62) = 62
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,55) = 55
write(1,"DEBUG: Player 2 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship can be placed here\n",31) = 31
write(1,"DEBUG: Ship placed successfully."...,62) = 62
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,55) = 55
write(1,"DEBUG: Player 2 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship can be placed here\n",31) = 31
write(1,"DEBUG: Ship placed successfully."...,62) = 62
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,55) = 55
write(1,"DEBUG: Player 2 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship can be placed here\n",31) = 31
write(1,"DEBUG: Ship placed successfully."...,62) = 62
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
```

```
write(1,"DEBUG: Received PLACE_SHIP from "...,55) = 55
write(1,"DEBUG: Player 2 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship can be placed here\n",31) = 31
write(1,"DEBUG: Ship placed successfully."...,62) = 62
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,55) = 55
write(1,"DEBUG: Player 2 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship placed successfully."...,62) = 62
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,55) = 55
write(1,"DEBUG: Player 2 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship placed successfully."...,62) = 62
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,55) = 55
write(1,"DEBUG: Player 2 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship placed successfully."...,62) = 62
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Received PLACE_SHIP from "...,55) = 55
write(1,"DEBUG: Player 2 placing ship\n",29) = 29
write(1,"DEBUG: Checking if can place shi"...,50) = 50
write(1,"DEBUG: Ship placed successfully."...,63) = 63
write(1,"DEBUG: Ship placed successfully\n",32) = 32
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAIT,2,NULL) = -1 EAGAIN (Resource temporarily unavailable
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
```

```
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAIT,2,NULL) = -1 EAGAIN (Resource temporarily unavailable
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAIT,2,NULL) = 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"DEBUG: Hit ship 9 at 9,0. Health"...,40) = 40
write(1,"DEBUG: Ship 9 SUNK! Marking cell"...,37) = 37
write(1,"DEBUG: Player 2 sunk a ship! Tot"...,43) = 43
write(1,"DEBUG: Hit but not sunk,shooter"...,51) = 51
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"Client quit: alice\n",19)    = 19
futex(0x7b5f05c84050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= 0
futex(0x7b5f05c84000,FUTEX_WAKE,1)    = 0
write(1,"Client quit: dmitriy\n",21)  = 21
futex(0x7b5f05c84054,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
= ? ERESTARTSYS (To be restarted if SA_RESTART is set)
---SIGINT {si_signo=SIGINT,si_code=SI_KERNEL} ---
+++ killed by SIGINT +++
```

Системные вызовы 1 игрока

```
execve("./client/client",["./client/client"],0x7ffdaa1cfba0 /* 41 vars */)
= 0
brk(NULL)                                 = 0x63a6f8b27000
mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x7be7821e6000
access("/etc/ld.so.preload",R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC) = 3
fstat(3,{st_mode=S_IFREG|0644,st_size=34331,...}) = 0
mmap(NULL,34331,PROT_READ,MAP_PRIVATE,3,0) = 0x7be7821dd000
close(3)                                  = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libstdc++.so.6",O_RDONLY|O_CLOEXEC)
```

```
= 3
read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"...,832)
= 832
fstat(3,{st_mode=S_IFREG|0644,st_size=2592224,...}) = 0
mmap(NULL,2609472,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7be781e00000
mmap(0x7be781e9d000,1343488,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3
= 0x7be781e9d000
mmap(0x7be781fe5000,552960,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x1e5000)
= 0x7be781fe5000
mmap(0x7be78206c000,57344,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7be78206c000
mmap(0x7be78207a000,12608,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,-1
= 0x7be78207a000
close(3)                                = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libgcc_s.so.1",O_RDONLY|O_CLOEXEC) =
3
read(3,"\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"...,832)
= 832
fstat(3,{st_mode=S_IFREG|0644,st_size=183024,...}) = 0
mmap(NULL,185256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7be7821af000
mmap(0x7be7821b3000,147456,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7be7821b3000
mmap(0x7be7821d7000,16384,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x28000)
= 0x7be7821d7000
mmap(0x7be7821db000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0
= 0x7be7821db000
close(3)                                = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libc.so.6",O_RDONLY|O_CLOEXEC) = 3
read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"...,832)
= 832
pread64(3,"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"...,784,64)
= 784
fstat(3,{st_mode=S_IFREG|0755,st_size=2125328,...}) = 0
pread64(3,"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"...,784,64)
= 784
mmap(NULL,2170256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7be781a00000
mmap(0x7be781a28000,1605632,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3
= 0x7be781a28000
mmap(0x7be781bb0000,323584,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x1b0000)
= 0x7be781bb0000
mmap(0x7be781bff000,24576,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7be781bff000
mmap(0x7be781c05000,52624,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,-1
= 0x7be781c05000
close(3)                                = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libm.so.6",O_RDONLY|O_CLOEXEC) = 3
read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"...,832)
= 832
```

```
fstat(3,{st_mode=S_IFREG|0644,st_size=952616,...}) = 0
mmap(NULL,950296,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7be7820c6000
mmap(0x7be7820d6000,520192,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7be7820d6000
mmap(0x7be782155000,360448,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x8f000)
= 0x7be782155000
mmap(0x7be7821ad000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0
= 0x7be7821ad000
close(3)                                = 0
mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x7be7820c4000
mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x7be7820c1000
arch_prctl(ARCH_SET_FS,0x7be7820c1740) = 0
set_tid_address(0x7be7820c1a10)         = 57149
set_robust_list(0x7be7820c1a20,24)      = 0
rseq(0x7be7820c2060,0x20,0,0x53053053) = 0
mprotect(0x7be781bff000,16384,PROT_READ) = 0
mprotect(0x7be7821ad000,4096,PROT_READ) = 0
mprotect(0x7be7821db000,4096,PROT_READ) = 0
mprotect(0x7be78206c000,45056,PROT_READ) = 0
mprotect(0x63a6d0ddb000,4096,PROT_READ) = 0
mprotect(0x7be78221e000,8192,PROT_READ) = 0
prlimit64(0,RLIMIT_STACK,NULL,{rlim_cur=8192*1024,rlim_max=RLIM64_INFINITY})
= 0
munmap(0x7be7821dd000,34331)            = 0
futex(0x7be78207a7bc,FUTEX_WAKE_PRIVATE,2147483647) = 0
getrandom("\xa4\x0f\x4d\xd0\x12\x49\xe2\x40",8,GRND_NONBLOCK) = 8
brk(NULL)                               = 0x63a6f8b27000
brk(0x63a6f8b48000)                     = 0x63a6f8b48000
openat(AT_FDCWD,"/dev/shm/battleship_shm_v3",O_RDWR|O_NOFOLLOW|O_CLOEXEC) =
3
mmap(NULL,68720,PROT_READ|PROT_WRITE,MAP_SHARED,3,0) = 0x7be7820b0000
fstat(1,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x4),...}) = 0
write(1,"================================"...,51) = 51
write(1,"  \320\224\320\236\320\221\320\240\320\236 \320\237\320\236\320\226\320\220\
= 60
write(1,"================================"...,51) = 51
write(1,"  \320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\262\320\260\
\320\273\320\276\320\263\320\270"...,36) = 36
fstat(0,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x4),...}) = 0
read(0,"dmitriy\n",1024)                = 8
write(1,"\n\360\237\224\227 \320\240\320\265\320\263\320\270\321\201\321\202\321\200\
= 32
futex(0x7be7820b0050,FUTEX_WAKE,1)      = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"\342\234\205 OK\n",7)          = 7
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
```

```
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"================================"...,51) = 51
write(1,"  \320\234\320\236\320\240\320\241\320\232\320\236\320\231 \320\221\320\236\
= 24
write(1,"================================"...,51) = 51
write(1,"  1 -\320\241\320\277\320\270\321\201\320\276\320\272 \320\270\320\263\321\2
= 44
write(1,"  2 -\320\241\320\276\320\267\320\264\320\260\321\202\321\214 \320\277\321\2
= 49
write(1,"  3 -\320\237\321\200\320\270\321\201\320\276\320\265\320\264\320\270\320\27
= 47
write(1,"  4 -\320\237\321\200\320\270\320\263\320\273\320\260\321\201\320\270\321\20
\320\270\320\263\321"...,40) = 40
write(1,"  5 -\320\237\321\200\320\276\320\262\320\265\321\200\320\270\321\202\321\21
\320\277\321\200\320\270\320"...,48) = 48
write(1,"  6 -\320\222\321\213\320\271\321\202\320\270\n",17) = 17
write(1,"  \320\222\321\213\320\261\320\265\321\200\320\270\321\202\320\265
\320\264\320\265\320\271\321\201\321\202\320\262\320"...,37) = 37
read(0,"2\n",1024)                      = 2
write(1,"\n",1)                         = 1
write(1,"\360\237\216\256 \320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\270\320\274\321\217 \320\264\320\273\321"...,55) = 55
read(0,"game1\n",1024)                  = 6
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"\342\234\205 \320\237\321\203\320\261\320\273\320\270\321\207\320\275\320\26
\320\270\320\263\321\200\320\260 "...,63) = 63
write(1,"\n",1)                         = 1
write(1,"\342\234\205 \320\237\321\203\320\261\320\273\320\270\321\207\320\275\320\26
\320\270\320\263\321\200\320\260 "...,63) = 63
write(1,"\n",1)                         = 1
write(1,"GAME_CREATED:\320\237\321\203\320\261\320\273\320\270\321\207\320\275\320\26
"...,72) = 72
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"================================"...,51) = 51
write(1,"  \320\240\320\220\320\241\320\241\320\242\320\220\320\235\320\236\320\222\3
```

```
\320\232\320\236\320\240\320"...,42) = 42
write(1,"================================"...,51) = 51
write(1,"  \320\244\320\276\321\200\320\274\320\260\321\202: \321\200\320\260\320\267
= 59
write(1,"  \320\237\321\200\320\270\320\274\320\265\321\200: 4,0,0,H\n\n",25)
= 25
write(1,"  \320\232\320\276\321\200\320\260\320\261\320\273\320\270 \320\264\320\273\
\321\200\320\260\320\267\320\274"...,46) = 46
write(1,"    1 \320\260\320\262\320\270\320\260\320\275\320\276\321\201\320\265\321\2
(4 \320\272\320\273"...,42) = 42
write(1,"    2 \320\273\320\270\320\275\320\272\320\276\321\200\320\260 (3
\320\272\320\273\320\265\321\202"...,38) = 38
write(1,"    3 \320\272\321\200\320\265\320\271\321\201\320\265\321\200\320\260
(2 \320\272\320\273\320\265"...,40) = 40
write(1,"    4 \321\215\321\201\320\274\320\270\320\275\321\206\320\260 (1
\320\272\320\273\320\265\321\202"...,38) = 38
write(1,"--------------------------------"...,51) = 51
write(1,"  \320\232\320\276\320\274\320\260\320\275\320\264\321\213:\n",18)
= 18
write(1,"    auto -\320\260\320\262\321\202\320\276\320\274\320\260\321\202\320\270\3
= 63
write(1,"    ready -\320\263\320\276\321\202\320\276\320\262 \320\272 \320\270\320\26
= 35
write(1,"    board -\320\277\320\276\321\201\320\274\320\276\321\202\321\200\320\265\
= 42
write(1,"    invite <\320\273\320\276\320\263\320\270\320\275>-\320\277\321\200\320\2
= 79
write(1,"    menu -\320\262\321\213\320\271\321\202\320\270 \320\262 \320\274\320\265
= 34
write(1,"--------------------------------"...,51) = 51
write(1,"\n",1)                         = 1
write(1,"\342\232\223 \320\232\320\276\320\274\320\260\320\275\320\264\320\260:
",20) = 20
read(0,"invite alice\n",1024)         = 13
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"\342\234\205 \320\237\321\200\320\270\320\263\320\273\320\260\321\210\320\26
\320\276\321\202\320"...,48) = 48
write(1,"\360\237\216\256 \320\222\321\213 \320\262\320\276\321\210\320\273\320\270
\320\262 \320\270\320\263\321\200\321\203"...,94) = 94
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
```

```
write(1,"\n",1)                        = 1
write(1,"================================"...,51) = 51
write(1,"  \320\240\320\220\320\241\320\241\320\242\320\220\320\235\320\236\320\222\3
\320\232\320\236\320\240\320"...,42) = 42
write(1,"================================"...,51) = 51
write(1,"  \320\244\320\276\321\200\320\274\320\260\321\202: \321\200\320\260\320\267
= 59
write(1,"  \320\237\321\200\320\270\320\274\320\265\321\200: 4,0,0,H\n\n",25)
= 25
write(1,"  \320\232\320\276\321\200\320\260\320\261\320\273\320\270 \320\264\320\273\3
\321\200\320\260\320\267\320\274"...,46) = 46
write(1,"    1 \320\260\320\262\320\270\320\260\320\275\320\276\321\201\320\265\321\20
(4 \320\272\320\273"...,42) = 42
write(1,"    2 \320\273\320\270\320\275\320\272\320\276\321\200\320\260 (3
\320\272\320\273\320\265\321\202"...,38) = 38
write(1,"    3 \320\272\321\200\320\265\320\271\321\201\320\265\321\200\320\260
(2 \320\272\320\273\320\265"...,40) = 40
write(1,"    4 \321\215\321\201\320\274\320\270\320\275\321\206\320\260 (1
\320\272\320\273\320\265\321\202"...,38) = 38
write(1,"--------------------------------"...,51) = 51
write(1,"  \320\232\320\276\320\274\320\260\320\275\320\264\321\213:\n",18)
= 18
write(1,"    auto -\320\260\320\262\321\202\320\276\320\274\320\260\321\202\320\270\3
= 63
write(1,"    ready -\320\263\320\276\321\202\320\276\320\262 \320\272 \320\270\320\26
= 35
write(1,"    board -\320\277\320\276\321\201\320\274\320\276\321\202\321\200\320\265\3
= 42
write(1,"    invite <\320\273\320\276\320\263\320\270\320\275>-\320\277\321\200\320\27
= 79
write(1,"    menu -\320\262\321\213\320\271\321\202\320\270 \320\262 \320\274\320\265
= 34
write(1,"--------------------------------"...,51) = 51
write(1,"\n",1)                        = 1
write(1,"\342\232\223 \320\232\320\276\320\274\320\260\320\275\320\264\320\260:
",20) = 20
read(0,"auto\n",1024)                  = 5
write(1,"[DEBUG] Buffer has: SHIP_PLACEME"...,185) = 185
write(1,"\320\232\320\276\320\263\320\264\320\260 \320\263\320\276\321\202\320\276\32
ready\n",31) = 31
write(1,"[DEBUG] Clearing buffer\n",24) = 24
write(1,"\n\360\237\224\204 \320\227\320\260\320\277\321\203\321\201\320\272\320\260\3
\320\260\320\262\321\202\320"...,80) = 80
write(1,"\n",1)                        = 1
write(1,"================================"...,51) = 51
write(1,"  \320\220\320\222\320\242\320\236\320\234\320\220\320\242\320\230\320\247\3
\320"...,71) = 71
write(1,"================================"...,51) = 51
```

```
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
futex(0x7be7820b0000,FUTEX_WAKE,1)    = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
futex(0x7be7820b0000,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
```

```
futex(0x7be7820b0000,FUTEX_WAKE,1)    = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                        = 1
write(1,"--------------------------------"...,51) = 51
write(1,"  \320\240\320\220\320\241\320\241\320\242\320\220\320\235\320\236\320\222\3
\320\227\320\220\320\222\320"...,44) = 44
write(1,"  \320\240\320\260\320\267\320\274\320\265\321\211\320\265\320\275\320\276
\320\272\320\276\321\200\320\260\320\261\320"...,45) = 45
write(1,"  \342\234\205 \320\222\321\201\320\265 \320\272\320\276\321\200\320\260\320
\321\203\321\201"...,63) = 63
write(1,"  \320\237\320\276\320\272\320\260\320\267\321\213\320\262\320\260\320\265\3
\320\277\320\276\320\273\320\265."...,35) = 35
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
futex(0x7be7820b0000,FUTEX_WAKE,1)    = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                        = 1
write(1,"YOUR_BOARD:\n    0  1  2  3  4  5"...,386) = 386
write(1,"\n",1)                        = 1
write(1,"  \320\224\320\273\321\217 \320\267\320\260\320\262\320\265\321\200\321\210\
\321\200"...,76) = 76
write(1,"================================"...,52) = 52
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                        = 1
write(1,"================================"...,51) = 51
write(1,"  \320\240\320\220\320\241\320\241\320\242\320\220\320\235\320\236\320\222\3
\320\232\320\236\320\240\320"...,42) = 42
write(1,"================================"...,51) = 51
write(1,"  \320\244\320\276\321\200\320\274\320\260\321\202: \321\200\320\260\320\267
= 59
write(1,"  \320\237\321\200\320\270\320\274\320\265\321\200: 4,0,0,H\n\n",25)
= 25
write(1,"  \320\232\320\276\321\200\320\260\320\261\320\273\320\270 \320\264\320\273\
\321\200\320\260\320\267\320\274"...,46) = 46
write(1,"    1 \320\260\320\262\320\270\320\260\320\275\320\276\321\201\320\265\321\2
(4 \320\272\320\273"...,42) = 42
write(1,"    2 \320\273\320\270\320\275\320\272\320\276\321\200\320\260 (3
\320\272\320\273\320\265\321\202"...,38) = 38
write(1,"    3 \320\272\321\200\320\265\320\271\321\201\320\265\321\200\320\260
(2 \320\272\320\273\320\265"...,40) = 40
```

```
write(1,"    4 \321\215\321\201\320\274\320\270\320\275\321\206\320\260 (1
\320\272\320\273\320\265\321\202"...,38) = 38
write(1,"--------------------------------"...,51) = 51
write(1,"  \320\232\320\276\320\274\320\260\320\275\320\264\321\213:\n",18)
= 18
write(1,"    auto -\320\260\320\262\321\202\320\276\320\274\320\260\321\202\320\270\3
= 63
write(1,"    ready -\320\263\320\276\321\202\320\276\320\262 \320\272 \320\270\320\26
= 35
write(1,"    board -\320\277\320\276\321\201\320\274\320\276\321\202\321\200\320\265\
= 42
write(1,"    invite <\320\273\320\276\320\263\320\270\320\275>-\320\277\321\200\320\2
= 79
write(1,"    menu -\320\262\321\213\320\271\321\202\320\270 \320\262 \320\274\320\265
= 34
write(1,"--------------------------------"...,51) = 51
write(1,"\n",1)                         = 1
write(1,"\342\232\223 \320\232\320\276\320\274\320\260\320\275\320\264\320\260:
",20) = 20
read(0,"ready\n",1024)                  = 6
write(1,"\360\237\224\204 \320\236\321\202\320\277\321\200\320\260\320\262\320\273\32
'ready"...,55) = 55
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
futex(0x7be7820b0000,FUTEX_WAKE,1)    = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\360\237\223\245 \320\237\320\276\320\273\321\203\321\207\320\265\320\275
\320\276\321\202\320\262\320\265\321\202 \320"...,51) = 51
write(1,"\n",1)                         = 1
write(1,"\342\234\205 Waiting for opponent...\n",28) = 28
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"================================"...,41) = 41
write(1,"  \320\230\320\223\320\240\320\220 \320\222 \320\237\320\240\320\236\320\246
= 31
write(1,"================================"...,41) = 41
write(1,"  1 -\320\241\320\264\320\265\320\273\320\260\321\202\321\214 \320\262\321\2
= 36
write(1,"  2 -\320\237\320\276\321\201\320\274\320\276\321\202\321\200\320\265\321\20
\321\201\320\262\320"...,45) = 45
write(1,"  3 -\320\237\320\276\321\201\320\274\320\276\321\202\321\200\320\265\321\20
\320\277\320\276\320"...,57) = 57
write(1,"  4 -\320\241\321\202\320\260\321\202\321\203\321\201 \320\270\320\263\321\2
```

```
= 28
write(1,"  5 -\320\241\320\264\320\260\321\202\321\214\321\201\321\217\n",21)
= 21
write(1,"  6 -\320\222\321\213\320\271\321\202\320\270 \320\262 \320\274\320\265\320\
= 29
write(1,"--------------------------------"...,41) = 41
write(1,"  \320\222\321\213\320\261\320\265\321\200\320\270\321\202\320\265
\320\264\320\265\320\271\321\201\321\202\320\262\320"...,37) = 37
read(0,"4\n",1024)                      = 2
futex(0x7be7820b0054,FUTEX_WAKE,1)      = 1
futex(0x7be7820b0000,FUTEX_WAKE,1)      = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"\n\320\230\320\263\321\200\320\260: game1 (ID: 0)\n\320\230\320\263\321\200\
= 285
write(1,"\n",1)                         = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"================================"...,41) = 41
write(1,"  \320\230\320\223\320\240\320\220 \320\222 \320\237\320\240\320\236\320\246
= 31
write(1,"================================"...,41) = 41
write(1,"  1 -\320\241\320\264\320\265\320\273\320\260\321\202\321\214 \320\262\321\2
= 36
write(1,"  2 -\320\237\320\276\321\201\320\274\320\276\321\202\321\200\320\265\321\20
\321\201\320\262\320"...,45) = 45
write(1,"  3 -\320\237\320\276\321\201\320\274\320\276\321\202\321\200\320\265\321\20
\320\277\320\276\320"...,57) = 57
write(1,"  4 -\320\241\321\202\320\260\321\202\321\203\321\201 \320\270\320\263\321\2
= 28
write(1,"  5 -\320\241\320\264\320\260\321\202\321\214\321\201\321\217\n",21)
= 21
write(1,"  6 -\320\222\321\213\320\271\321\202\320\270 \320\262 \320\274\320\265\320\
= 29
write(1,"--------------------------------"...,41) = 41
write(1,"  \320\222\321\213\320\261\320\265\321\200\320\270\321\202\320\265
\320\264\320\265\320\271\321\201\321\202\320\262\320"...,37) = 37
read(0,"1\n",1024)                      = 2
write(1,"\n",1)                         = 1
write(1,"\360\237\216\257 \320\232\320\276\320\276\321\200\320\264\320\270\320\275\32
\320\262\321\213\321\201"...,50) = 50
read(0,"7,7\n",1024)                    = 4
```

```
write(1,"\360\237\224\204 \320\236\321\202\320\277\321\200\320\260\320\262\320\273\32
\320\262\321\213\321\201"...,44) = 44
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\360\237\223\245 \320\236\321\202\320\262\320\265\321\202 \321\201\320\265\32
\320"...,46) = 46
write(1,"\n",1)                       = 1
write(1,"\n    0  1  2  3  4  5  6  7  8  "...,375) = 375
write(1,"\n",1)                       = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                       = 1
write(1,"================================"...,41) = 41
write(1,"  \320\230\320\223\320\240\320\220 \320\222 \320\237\320\240\320\236\320\246
= 31
write(1,"================================"...,41) = 41
write(1,"  1 -\320\241\320\264\320\265\320\273\320\260\321\202\321\214 \320\262\321\2
= 36
write(1,"  2 -\320\237\320\276\321\201\320\274\320\276\321\202\321\200\320\265\321\20
\321\201\320\262\320"...,45) = 45
write(1,"  3 -\320\237\320\276\321\201\320\274\320\276\321\202\321\200\320\265\321\20
\320\277\320\276\320"...,57) = 57
write(1,"  4 -\320\241\321\202\320\260\321\202\321\203\321\201 \320\270\320\263\321\2
= 28
write(1,"  5 -\320\241\320\264\320\260\321\202\321\214\321\201\321\217\n",21)
= 21
write(1,"  6 -\320\222\321\213\320\271\321\202\320\270 \320\262 \320\274\320\265\320\
= 29
write(1,"--------------------------------"...,41) = 41
write(1,"  \320\222\321\213\320\261\320\265\321\200\320\270\321\202\320\265
\320\264\320\265\320\271\321\201\321\202\320\262\320"...,37) = 37
read(0,"5\n",1024)                    = 2
write(1,"\360\237\224\204 \320\237\321\200\320\276\320\262\320\265\321\200\321\217\32
\321\201\320\276\321\201\321\202"...,55) = 55
write(1,"[DEBUG] Buffer has: OPPONENT_SHO"...,51) = 51
write(1,"[DEBUG] Clearing buffer\n",24) = 24
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\342\234\205 \320\230\320\263\321\200\320\260 \321\201\321\203\321\211\320\2
= 72
write(1,"\320\230\320\263\321\200\320\276\320\272 1:...\n",17) = 17
write(1,"\n",1)                       = 1
write(1,"\360\237\217\263\357\270\217 \320\222\321\213 \321\203\320\262\320\265\321\2
```

```
(\320\264"...,44) = 44
read(0,"\320\264\320\260\n",1024)     = 5
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                       = 1
write(1,"\360\237\227\221\357\270\217 \320\230\320\263\321\200\320\260 \321\203\320\2
= 32
write(1,"\n",1)                       = 1
write(1,"GAME_REMOVED:\320\230\320\263\321\200\320\260 \321\203\320\264\320\260\320\2
= 37
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
write(1,"\n",1)                       = 1
write(1,"================================"...,51) = 51
write(1,"  \320\234\320\236\320\240\320\241\320\232\320\236\320\231 \320\221\320\236\
= 24
write(1,"================================"...,51) = 51
write(1,"  1 -\320\241\320\277\320\270\321\201\320\276\320\272 \320\270\320\263\321\2
= 44
write(1,"  2 -\320\241\320\276\320\267\320\264\320\260\321\202\321\214 \320\277\321\2
= 49
write(1,"  3 -\320\237\321\200\320\270\321\201\320\276\320\265\320\264\320\270\320\27
= 47
write(1,"  4 -\320\237\321\200\320\270\320\263\320\273\320\260\321\201\320\270\321\20
\320\270\320\263\321"...,40) = 40
write(1,"  5 -\320\237\321\200\320\276\320\262\320\265\321\200\320\270\321\202\321\21
\320\277\321\200\320\270\320"...,48) = 48
write(1,"  6 -\320\222\321\213\320\271\321\202\320\270\n",17) = 17
write(1,"  \320\222\321\213\320\261\320\265\321\200\320\270\321\202\320\265
\320\264\320\265\320\271\321\201\321\202\320\262\320"...,37) = 37
read(0,"6\n",1024)                    = 2
write(1,"\n",1)                       = 1
write(1,"\360\237\232\252 \320\222\321\213 \321\203\320\262\320\265\321\200\320\265\3
(\320\264\320\260/"...,41) = 41
read(0,"\320\264\320\260\n",1024)     = 5
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
write(1,"\n\360\237\221\213 \320\222\321\213\321\205\320\276\320\264...\n",20)
= 20
write(1,"\n",1)                       = 1
write(1,"================================"...,51) = 51
write(1,"  \320\230\320\223\320\240\320\220 \320\227\320\220\320\222\320\225\320\240\
= 30
write(1,"  \320\241\320\277\320\260\321\201\320\270\320\261\320\276 \320\267\320\260
\320\270\320\263\321\200\321\203!\n",32) = 32
```

```
write(1,"================================"...,51) = 51
munmap(0x7be7820b0000,68720)           = 0
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Системные вызовы 2 игрока

```
execve("./client/client",["./client/client"],0x7ffdaa1cfba0 /* 41 vars */)
= 0
brk(NULL)                               = 0x63a6f8b27000
mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x7be7821e6000
access("/etc/ld.so.preload",R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC) = 3
fstat(3,{st_mode=S_IFREG|0644,st_size=34331,...}) = 0
mmap(NULL,34331,PROT_READ,MAP_PRIVATE,3,0) = 0x7be7821dd000
close(3)                                = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libstdc++.so.6",O_RDONLY|O_CLOEXEC)
= 3
read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"...,832)
= 832
fstat(3,{st_mode=S_IFREG|0644,st_size=2592224,...}) = 0
mmap(NULL,2609472,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7be781e00000
mmap(0x7be781e9d000,1343488,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3
= 0x7be781e9d000
mmap(0x7be781fe5000,552960,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x1e5000)
= 0x7be781fe5000
mmap(0x7be78206c000,57344,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7be78206c000
mmap(0x7be78207a000,12608,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,-1
= 0x7be78207a000
close(3)                                = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libgcc_s.so.1",O_RDONLY|O_CLOEXEC) =
3
read(3,"\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"...,832)
= 832
fstat(3,{st_mode=S_IFREG|0644,st_size=183024,...}) = 0
mmap(NULL,185256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7be7821af000
mmap(0x7be7821b3000,147456,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7be7821b3000
mmap(0x7be7821d7000,16384,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x28000)
= 0x7be7821d7000
mmap(0x7be7821db000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0
= 0x7be7821db000
close(3)                                = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libc.so.6",O_RDONLY|O_CLOEXEC) = 3
read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"...,832)
= 832
pread64(3,"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"...,784,64)
```

```
= 784
fstat(3,{st_mode=S_IFREG|0755,st_size=2125328,...}) = 0
pread64(3,"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"...,784,64)
= 784
mmap(NULL,2170256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7be781a00000
mmap(0x7be781a28000,1605632,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3
= 0x7be781a28000
mmap(0x7be781bb0000,323584,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x1b0000)
= 0x7be781bb0000
mmap(0x7be781bff000,24576,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7be781bff000
mmap(0x7be781c05000,52624,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,-1
= 0x7be781c05000
close(3)                                = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libm.so.6",O_RDONLY|O_CLOEXEC) = 3
read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"...,832)
= 832
fstat(3,{st_mode=S_IFREG|0644,st_size=952616,...}) = 0
mmap(NULL,950296,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7be7820c6000
mmap(0x7be7820d6000,520192,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7be7820d6000
mmap(0x7be782155000,360448,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x8f000)
= 0x7be782155000
mmap(0x7be7821ad000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0
= 0x7be7821ad000
close(3)                                = 0
mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x7be7820c4000
mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x7be7820c1000
arch_prctl(ARCH_SET_FS,0x7be7820c1740) = 0
set_tid_address(0x7be7820c1a10)         = 57149
set_robust_list(0x7be7820c1a20,24)      = 0
rseq(0x7be7820c2060,0x20,0,0x53053053) = 0
mprotect(0x7be781bff000,16384,PROT_READ) = 0
mprotect(0x7be7821ad000,4096,PROT_READ) = 0
mprotect(0x7be7821db000,4096,PROT_READ) = 0
mprotect(0x7be78206c000,45056,PROT_READ) = 0
mprotect(0x63a6d0ddb000,4096,PROT_READ) = 0
mprotect(0x7be78221e000,8192,PROT_READ) = 0
prlimit64(0,RLIMIT_STACK,NULL,{rlim_cur=8192*1024,rlim_max=RLIM64_INFINITY})
= 0
munmap(0x7be7821dd000,34331)            = 0
futex(0x7be78207a7bc,FUTEX_WAKE_PRIVATE,2147483647) = 0
getrandom("\xa4\x0f\x4d\xd0\x12\x49\xe2\x40",8,GRND_NONBLOCK) = 8
brk(NULL)                               = 0x63a6f8b27000
brk(0x63a6f8b48000)                     = 0x63a6f8b48000
openat(AT_FDCWD,"/dev/shm/battleship_shm_v3",O_RDWR|O_NOFOLLOW|O_CLOEXEC) =
3
mmap(NULL,68720,PROT_READ|PROT_WRITE,MAP_SHARED,3,0) = 0x7be7820b0000
```

```
fstat(1,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x4),...}) = 0
write(1,"=================================="...,51) = 51
write(1,"  \320\224\320\236\320\221\320\240\320\236 \320\237\320\236\320\226\320\220\
= 60
write(1,"=================================="...,51) = 51
write(1,"  \320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\262\320\260\
\320\273\320\276\320\263\320\270"...,36) = 36
fstat(0,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x4),...}) = 0
read(0,"dmitriy\n",1024)            = 8
write(1,"\n\360\237\224\227 \320\240\320\265\320\263\320\270\321\201\321\202\321\200\
= 32
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                       = 1
write(1,"\342\234\205 OK\n",7)        = 7
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
write(1,"\n",1)                       = 1
write(1,"=================================="...,51) = 51
write(1,"  \320\234\320\236\320\240\320\241\320\232\320\236\320\231 \320\221\320\236\
= 24
write(1,"=================================="...,51) = 51
write(1,"  1 -\320\241\320\277\320\270\321\201\320\276\320\272 \320\270\320\263\321\2
= 44
write(1,"  2 -\320\241\320\276\320\267\320\264\320\260\321\202\321\214 \320\277\321\2
= 49
write(1,"  3 -\320\237\321\200\320\270\321\201\320\276\320\265\320\264\320\270\320\27
= 47
write(1,"  4 -\320\237\321\200\320\270\320\263\320\273\320\260\321\201\320\270\321\20
\320\270\320\263\321"...,40) = 40
write(1,"  5 -\320\237\321\200\320\276\320\262\320\265\321\200\320\270\321\202\321\21
\320\277\321\200\320\270\320"...,48) = 48
write(1,"  6 -\320\222\321\213\320\271\321\202\320\270\n",17) = 17
write(1,"  \320\222\321\213\320\261\320\265\321\200\320\270\321\202\320\265
\320\264\320\265\320\271\321\201\321\202\320\262\320"...,37) = 37
read(0,"2\n",1024)                    = 2
write(1,"\n",1)                       = 1
write(1,"\360\237\216\256 \320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\270\320\274\321\217 \320\264\320\273\321"...,55) = 55
read(0,"game1\n",1024)                = 6
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                       = 1
write(1,"\342\234\205 \320\237\321\203\320\261\320\273\320\270\321\207\320\275\320\26
```

```
\320\270\320\263\321\200\320\260 "...,63) = 63
write(1,"\n",1)                         = 1
write(1,"\342\234\205 \320\237\321\203\320\261\320\273\320\270\321\207\320\275\320\26
\320\270\320\263\321\200\320\260 "...,63) = 63
write(1,"\n",1)                         = 1
write(1,"GAME_CREATED:\320\237\321\203\320\261\320\273\320\270\321\207\320\275\320\26
"...,72) = 72
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"================================"...,51) = 51
write(1,"  \320\240\320\220\320\241\320\241\320\242\320\220\320\235\320\236\320\222\3
\320\232\320\236\320\240\320"...,42) = 42
write(1,"================================"...,51) = 51
write(1,"  \320\244\320\276\321\200\320\274\320\260\321\202: \321\200\320\260\320\267
= 59
write(1,"  \320\237\321\200\320\270\320\274\320\265\321\200: 4,0,0,H\n\n",25)
= 25
write(1,"  \320\232\320\276\321\200\320\260\320\261\320\273\320\270 \320\264\320\273\
\321\200\320\260\320\267\320\274"...,46) = 46
write(1,"    1 \320\260\320\262\320\270\320\260\320\275\320\276\321\201\320\265\321\20
(4 \320\272\320\273"...,42) = 42
write(1,"    2 \320\273\320\270\320\275\320\272\320\276\321\200\320\260 (3
\320\272\320\273\320\265\321\202"...,38) = 38
write(1,"    3 \320\272\321\200\320\265\320\271\321\201\320\265\321\200\320\260
(2 \320\272\320\273\320\265"...,40) = 40
write(1,"    4 \321\215\321\201\320\274\320\270\320\275\321\206\320\260 (1
\320\272\320\273\320\265\321\202"...,38) = 38
write(1,"--------------------------------"...,51) = 51
write(1,"  \320\232\320\276\320\274\320\260\320\275\320\264\321\213:\n",18)
= 18
write(1,"    auto -\320\260\320\262\321\202\320\276\320\274\320\260\321\202\320\270\3
= 63
write(1,"    ready -\320\263\320\276\321\202\320\276\320\262 \320\272 \320\270\320\26
= 35
write(1,"    board -\320\277\320\276\321\201\320\274\320\276\321\202\321\200\320\265\
= 42
write(1,"    invite <\320\273\320\276\320\263\320\270\320\275>-\320\277\321\200\320\2
= 79
write(1,"    menu -\320\262\321\213\320\271\321\202\320\270 \320\262 \320\274\320\265
= 34
write(1,"--------------------------------"...,51) = 51
write(1,"\n",1)                         = 1
```

```
write(1,"\342\232\223 \320\232\320\276\320\274\320\260\320\275\320\264\320\260:
",20) = 20
read(0,"invite alice\n",1024)         = 13
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                       = 1
write(1,"\342\234\205 \320\237\321\200\320\270\320\263\320\273\320\260\321\210\320\26
\320\276\321\202\320"...,48) = 48
write(1,"\360\237\216\256 \320\222\321\213 \320\262\320\276\321\210\320\273\320\270
\320\262 \320\270\320\263\321\200\321\203"...,94) = 94
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                       = 1
write(1,"================================"...,51) = 51
write(1,"  \320\240\320\220\320\241\320\241\320\242\320\220\320\235\320\236\320\222\3
\320\232\320\236\320\240\320"...,42) = 42
write(1,"================================"...,51) = 51
write(1,"  \320\244\320\276\321\200\320\274\320\260\321\202: \321\200\320\260\320\267
= 59
write(1,"  \320\237\321\200\320\270\320\274\320\265\321\200: 4,0,0,H\n\n",25)
= 25
write(1,"  \320\232\320\276\321\200\320\260\320\261\320\273\320\270 \320\264\320\273\
\321\200\320\260\320\267\320\274"...,46) = 46
write(1,"    1 \320\260\320\262\320\270\320\260\320\275\320\276\321\201\320\265\321\2
(4 \320\272\320\273"...,42) = 42
write(1,"    2 \320\273\320\270\320\275\320\272\320\276\321\200\320\260 (3
\320\272\320\273\320\265\321\202"...,38) = 38
write(1,"    3 \320\272\321\200\320\265\320\271\321\201\320\265\321\200\320\260
(2 \320\272\320\273\320\265"...,40) = 40
write(1,"    4 \321\215\321\201\320\274\320\270\320\275\321\206\320\260 (1
\320\272\320\273\320\265\321\202"...,38) = 38
write(1,"--------------------------------"...,51) = 51
write(1,"  \320\232\320\276\320\274\320\260\320\275\320\264\321\213:\n",18)
= 18
write(1,"    auto -\320\260\320\262\321\202\320\276\320\274\320\260\321\202\320\270\3
= 63
write(1,"    ready -\320\263\320\276\321\202\320\276\320\262 \320\272 \320\270\320\26
= 35
write(1,"    board -\320\277\320\276\321\201\320\274\320\276\321\202\321\200\320\265\
= 42
write(1,"    invite <\320\273\320\276\320\263\320\270\320\275>-\320\277\321\200\320\2
= 79
write(1,"    menu -\320\262\321\213\320\271\321\202\320\270 \320\262 \320\274\320\265
```

```
= 34
write(1,"--------------------------------"...,51) = 51
write(1,"\n",1)                         = 1
write(1,"\342\232\223 \320\232\320\276\320\274\320\260\320\275\320\264\320\260:
",20) = 20
read(0,"auto\n",1024)                   = 5
write(1,"[DEBUG] Buffer has: SHIP_PLACEME"...,185) = 185
write(1,"\320\232\320\276\320\263\320\264\320\260 \320\263\320\276\321\202\320\276\32
ready\n",31) = 31
write(1,"[DEBUG] Clearing buffer\n",24) = 24
write(1,"\n\360\237\224\204 \320\227\320\260\320\277\321\203\321\201\320\272\320\260\
\320\260\320\262\321\202\320"...,80) = 80
write(1,"\n",1)                         = 1
write(1,"================================"...,51) = 51
write(1,"  \320\220\320\222\320\242\320\236\320\234\320\220\320\242\320\230\320\247\3
\320"...,71) = 71
write(1,"================================"...,51) = 51
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
```

```
futex(0x7be7820b0000,FUTEX_WAKE,1)    = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
futex(0x7be7820b0000,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
futex(0x7be7820b0000,FUTEX_WAKE,1)    = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\320\240\320\260\320\267\320\274\320\265\321\211\320\260\320\265\320\274
\320\272\320\276\321\200\320\260\320\261\320\273\321"...,65) = 65
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"--------------------------------"...,51) = 51
write(1,"  \320\240\320\220\320\241\320\241\320\242\320\220\320\235\320\236\320\222\3
\320\227\320\220\320\222\320"...,44) = 44
write(1,"  \320\240\320\260\320\267\320\274\320\265\321\211\320\265\320\275\320\276
\320\272\320\276\321\200\320\260\320\261\320"...,45) = 45
write(1,"  \342\234\205 \320\222\321\201\320\265 \320\272\320\276\321\200\320\260\320
\321\203\321\201"...,63) = 63
write(1,"  \320\237\320\276\320\272\320\260\320\267\321\213\320\262\320\260\320\265\3
\320\277\320\276\320\273\320\265."...,35) = 35
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
futex(0x7be7820b0000,FUTEX_WAKE,1)    = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"YOUR_BOARD:\n    0  1  2  3  4  5"...,386) = 386
write(1,"\n",1)                         = 1
write(1,"  \320\224\320\273\321\217 \320\267\320\260\320\262\320\265\321\200\321\210\
\321\200"...,76) = 76
write(1,"================================"...,52) = 52
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
```

```
write(1,"\n",1)                          = 1
write(1,"================================"...,51) = 51
write(1,"  \320\240\320\220\320\241\320\241\320\242\320\220\320\235\320\236\320\222\3
\320\232\320\236\320\240\320"...,42) = 42
write(1,"================================"...,51) = 51
write(1,"  \320\244\320\276\321\200\320\274\320\260\321\202: \321\200\320\260\320\267
= 59
write(1,"  \320\237\321\200\320\270\320\274\320\265\321\200: 4,0,0,H\n\n",25)
= 25
write(1,"  \320\232\320\276\321\200\320\260\320\261\320\273\320\270 \320\264\320\273\
\321\200\320\260\320\267\320\274"...,46) = 46
write(1,"    1 \320\260\320\262\320\270\320\260\320\275\320\276\321\201\320\265\321\2
(4 \320\272\320\273"...,42) = 42
write(1,"    2 \320\273\320\270\320\275\320\272\320\276\321\200\320\260 (3
\320\272\320\273\320\265\321\202"...,38) = 38
write(1,"    3 \320\272\321\200\320\265\320\271\321\201\320\265\321\200\320\260
(2 \320\272\320\273\320\265"...,40) = 40
write(1,"    4 \321\215\321\201\320\274\320\270\320\275\321\206\320\260 (1
\320\272\320\273\320\265\321\202"...,38) = 38
write(1,"--------------------------------"...,51) = 51
write(1,"  \320\232\320\276\320\274\320\260\320\275\320\264\321\213:\n",18)
= 18
write(1,"    auto -\320\260\320\262\321\202\320\276\320\274\320\260\321\202\320\270\3
= 63
write(1,"    ready -\320\263\320\276\321\202\320\276\320\262 \320\272 \320\270\320\26
= 35
write(1,"    board -\320\277\320\276\321\201\320\274\320\276\321\202\321\200\320\265\
= 42
write(1,"    invite <\320\273\320\276\320\263\320\270\320\275>-\320\277\321\200\320\2
= 79
write(1,"    menu -\320\262\321\213\320\271\321\202\320\270 \320\262 \320\274\320\265
= 34
write(1,"--------------------------------"...,51) = 51
write(1,"\n",1)                          = 1
write(1,"\342\232\223 \320\232\320\276\320\274\320\260\320\275\320\264\320\260:
",20) = 20
read(0,"ready\n",1024)                   = 6
write(1,"\360\237\224\204 \320\236\321\202\320\277\321\200\320\260\320\262\320\273\32
'ready"...,55) = 55
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
futex(0x7be7820b0000,FUTEX_WAKE,1)    = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\360\237\223\245 \320\237\320\276\320\273\321\203\321\207\320\265\320\275
\320\276\321\202\320\262\320\265\321\202 \320"...,51) = 51
write(1,"\n",1)                          = 1
write(1,"\342\234\205 Waiting for opponent...\n",28) = 28
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
```

```
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"================================"...,41) = 41
write(1,"  \320\230\320\223\320\240\320\220 \320\222 \320\237\320\240\320\236\320\246
= 31
write(1,"================================"...,41) = 41
write(1,"  1 -\320\241\320\264\320\265\320\273\320\260\321\202\321\214 \320\262\321\2
= 36
write(1,"  2 -\320\237\320\276\321\201\320\274\320\276\321\202\321\200\320\265\321\20
\321\201\320\262\320"...,45) = 45
write(1,"  3 -\320\237\320\276\321\201\320\274\320\276\321\202\321\200\320\265\321\20
\320\277\320\276\320"...,57) = 57
write(1,"  4 -\320\241\321\202\320\260\321\202\321\203\321\201 \320\270\320\263\321\2
= 28
write(1,"  5 -\320\241\320\264\320\260\321\202\321\214\321\201\321\217\n",21)
= 21
write(1,"  6 -\320\222\321\213\320\271\321\202\320\270 \320\262 \320\274\320\265\320\
= 29
write(1,"--------------------------------"...,41) = 41
write(1,"  \320\222\321\213\320\261\320\265\321\200\320\270\321\202\320\265
\320\264\320\265\320\271\321\201\321\202\320\262\320"...,37) = 37
read(0,"4\n",1024)                      = 2
futex(0x7be7820b0054,FUTEX_WAKE,1)      = 1
futex(0x7be7820b0000,FUTEX_WAKE,1)      = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"\n\320\230\320\263\321\200\320\260: game1 (ID: 0)\n\320\230\320\263\321\200\
= 285
write(1,"\n",1)                         = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"================================"...,41) = 41
write(1,"  \320\230\320\223\320\240\320\220 \320\222 \320\237\320\240\320\236\320\246
= 31
write(1,"================================"...,41) = 41
write(1,"  1 -\320\241\320\264\320\265\320\273\320\260\321\202\321\214 \320\262\321\2
= 36
write(1,"  2 -\320\237\320\276\321\201\320\274\320\276\321\202\321\200\320\265\321\20
```

```
\321\201\320\262\320"...,45) = 45
write(1,"  3 -\320\237\320\276\321\201\320\274\320\276\321\202\321\200\320\265\321\20
\320\277\320\276\320"...,57) = 57
write(1,"  4 -\320\241\321\202\320\260\321\202\321\203\321\201 \320\270\320\263\321\2
= 28
write(1,"  5 -\320\241\320\264\320\260\321\202\321\214\321\201\321\217\n",21)
= 21
write(1,"  6 -\320\222\321\213\320\271\321\202\320\270 \320\262 \320\274\320\265\320\
= 29
write(1,"--------------------------------"...,41) = 41
write(1,"  \320\222\321\213\320\261\320\265\321\200\320\270\321\202\320\265
\320\264\320\265\320\271\321\201\321\202\320\262\320"...,37) = 37
read(0,"1\n",1024)                      = 2
write(1,"\n",1)                         = 1
write(1,"\360\237\216\257 \320\232\320\276\320\276\321\200\320\264\320\270\320\275\32
\320\262\321\213\321\201"...,50) = 50
read(0,"7,7\n",1024)                    = 4
write(1,"\360\237\224\204 \320\236\321\202\320\277\321\200\320\260\320\262\320\273\32
\320\262\321\213\321\201"...,44) = 44
futex(0x7be7820b0050,FUTEX_WAKE,1)      = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\360\237\223\245 \320\236\321\202\320\262\320\265\321\202 \321\201\320\265\3
\320"...,46) = 46
write(1,"\n",1)                         = 1
write(1,"\n    0  1  2  3  4  5  6  7  8  "...,375) = 375
write(1,"\n",1)                         = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"================================"...,41) = 41
write(1,"  \320\230\320\223\320\240\320\220 \320\222 \320\237\320\240\320\236\320\246
= 31
write(1,"================================"...,41) = 41
write(1,"  1 -\320\241\320\264\320\265\320\273\320\260\321\202\321\214 \320\262\321\2
= 36
write(1,"  2 -\320\237\320\276\321\201\320\274\320\276\321\202\321\200\320\265\321\20
\321\201\320\262\320"...,45) = 45
write(1,"  3 -\320\237\320\276\321\201\320\274\320\276\321\202\321\200\320\265\321\20
\320\277\320\276\320"...,57) = 57
write(1,"  4 -\320\241\321\202\320\260\321\202\321\203\321\201 \320\270\320\263\321\2
= 28
write(1,"  5 -\320\241\320\264\320\260\321\202\321\214\321\201\321\217\n",21)
= 21
```

```
write(1,"  6 -\320\222\321\213\320\271\321\202\320\270 \320\262 \320\274\320\265\320\
= 29
write(1,"--------------------------------"...,41) = 41
write(1,"  \320\222\321\213\320\261\320\265\321\200\320\270\321\202\320\265
\320\264\320\265\320\271\321\201\321\202\320\262\320"...,37) = 37
read(0,"5\n",1024)                      = 2
write(1,"\360\237\224\204 \320\237\321\200\320\276\320\262\320\265\321\200\321\217\320
\321\201\320\276\321\201\321\202"...,55) = 55
write(1,"[DEBUG] Buffer has: OPPONENT_SHO"...,51) = 51
write(1,"[DEBUG] Clearing buffer\n",24) = 24
futex(0x7be7820b0050,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\342\234\205 \320\230\320\263\321\200\320\260 \321\201\321\203\321\211\320\2
= 72
write(1,"\320\230\320\263\321\200\320\276\320\272 1:...\n",17) = 17
write(1,"\n",1)                         = 1
write(1,"\360\237\217\263\357\270\217 \320\222\321\213 \321\203\320\262\320\265\321\2
(\320\264"...,44) = 44
read(0,"\320\264\320\260\n",1024)     = 5
futex(0x7be7820b0054,FUTEX_WAKE,1)    = 1
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"\360\237\227\221\357\270\217 \320\230\320\263\321\200\320\260 \321\203\320\2
= 32
write(1,"\n",1)                         = 1
write(1,"GAME_REMOVED:\320\230\320\263\321\200\320\260 \321\203\320\264\320\260\320\2
= 37
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=100000000},NULL) = 0
clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=50000000},NULL) = 0
write(1,"\n",1)                         = 1
write(1,"================================"...,51) = 51
write(1,"  \320\234\320\236\320\240\320\241\320\232\320\236\320\231 \320\221\320\236\
= 24
write(1,"================================"...,51) = 51
write(1,"  1 -\320\241\320\277\320\270\321\201\320\276\320\272 \320\270\320\263\321\2
= 44
write(1,"  2 -\320\241\320\276\320\267\320\264\320\260\321\202\321\214 \320\277\321\2
= 49
write(1,"  3 -\320\237\321\200\320\270\321\201\320\276\320\265\320\264\320\270\320\27
= 47
write(1,"  4 -\320\237\321\200\320\270\320\263\320\273\320\260\321\201\320\270\321\20
\320\270\320\263\321"...,40) = 40
write(1,"  5 -\320\237\321\200\320\276\320\262\320\265\321\200\320\270\321\202\321\21
\320\277\321\200\320\270\320"...,48) = 48
```

```
write(1,"  6 -\320\222\321\213\320\271\321\202\320\270\n",17) = 17
write(1,"  \320\222\321\213\320\261\320\265\321\200\320\270\321\202\320\265
\320\264\320\265\320\271\321\201\321\202\320\262\320"...,37) = 37
read(0,"6\n",1024)                      = 2
write(1,"\n",1)                         = 1
write(1,"\360\237\232\252 \320\222\321\213 \321\203\320\262\320\265\321\200\320\265\3
(\320\264\320\260/"...,41) = 41
read(0,"\320\264\320\260\n",1024)       = 5
futex(0x7be7820b0050,FUTEX_WAKE,1)      = 1
write(1,"\n\360\237\221\213 \320\222\321\213\321\205\320\276\320\264...\n",20)
= 20
write(1,"\n",1)                         = 1
write(1,"================================"...,51) = 51
write(1,"  \320\230\320\223\320\240\320\220 \320\227\320\220\320\222\320\225\320\240\
= 30
write(1,"  \320\241\320\277\320\260\321\201\320\270\320\261\320\276 \320\267\320\260
\320\270\320\263\321\200\321\203!\n",32) = 32
write(1,"================================"...,51) = 51
munmap(0x7be7820b0000,68720)             = 0
close(3)                                  = 0
exit_group(0)                             = ?
+++ exited with 0 +++
```