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»

Выполнила: В. А. Гузова
Группа: М8О-207БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Условие

Цель курсового проекта:

- Приобретение практических навыков в использовании знаний, полученных в течении курса
- 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. Проектирование консольной клиент-серверной игры На основе любой из выбранных технологий:

- Pipes
- Sockets
- Сервера очередей
- И другие

Создать собственную игру более, чем для одного пользователя. Игра может быть устроена по принципу: клиент-клиент, сервер-клиент.

Вариант: 1

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

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

Метод решения

Для реализации выбрана классическая архитектура «Клиент-Сервер» на базе IPC (Inter-Process Communication).Взаимодействие между процессами реализовано через именнованные каналы в ОС Linux.Используется библиотека POSIX Threads(для создания динамичной игры: клиент может видеть,что соперник совершил ход)

Алгоритм решения задачи

Запуск сервера

- Сервер создает главный именованный канал (FIFO) по фиксированному пути /tmp/sb_server_in с правами доступа 0666.
- Канал открывается в режиме чтения, после чего сервер входит в режим ожидания входящих соединений.
- Сервер входит в бесконечный цикл ожидания пакетов struct Packet фиксированного размера из главного канала.

Запуск клиента и авторизация

- При запуске клиент запрашивает логин пользователя.
- Создается уникальный именованный канал для приема сообщений от сервера: /tmp/sb_client_<login>.
- Запускается фоновый поток с использованием функции pthread_create, который открывает личный канал на чтение и блокируется в ожидании сообщений от сервера.
- Клиент отправляет пакет типа LOGIN в общий канал сервера, сообщая свой идентификатор для регистрации в системе.

Организация комнаты

- Один игрок выбирает «Create Game», отправляет имя комнаты. Сервер создает объект GameSession, где player1 - создатель, а player2 пока пуст.
- Второй игрок: Выбирает «Join Game» и вводит имя той же комнаты.
- Сервер находит комнату, прописывает второго игрока и отправляет обоим пакет JOIN_GAME. Теперь клиенты знают имена своих друзей (Friend)

Расстановка кораблей

- Как только комната собрана, клиент автоматически вызывает placeShipsRandomly().
- Алгоритм расстановки: Используется массив размеров кораблей 4, 3, 3.... Для каждого генерируются случайные координаты и направление. Проверяется отсутствие столкновений и «соседства» (дистанция в 1 клетку).
- Клиент отправляет готовое поле на сервер командой PLACE_SHIPS. Сервер сохраняет его в структуре Player.

Игровой процесс

- Для выстрела игрок вводит координаты. Клиент шлет пакет SHOT серверу.
- Сервер находит поле друга.Проверяет клетку: SHIP, EMPTY, RES_REPEAT. Обновляет поле жертвы.

- Если игрок попал, флаг your_turnостается true, и он стреляет снова. Если промах - ход переходит к другу.

Завершение игры

- После каждого хита сервер проверяет, остались ли у жертвы целые палубы (SHIP).
- Если все корабли уничтожены, сервер: Устанавливает результат RES_LOSE для проигравшего. Обновляет счетчик wins и losses в векторе игроков. Вызывает saveStatsToFile(), сохраняя новые данные на диск.
- Клиенты получают пакет завершения, выводят сообщение о победителе и возвращаются в меню.

Описание программы

Разделение по файлам, описание основных типов данных и функций. Обязательно написать используемые системные вызовы.

common.h — содержит общие структуры данных и константы для взимодействия клиента и сервера. Основные определения:

- enum RequestType - перечисление типов сообщений для управления протоколом обмена.
- struct Packet - структура фиксируемого размера,используется для передачи данных через каналы.
- struct Player - структура,с помощью которой хранятся данные игрока на сервере.

pipes.h — обертка над системными вызовами для работы с именнованными каналами. Основные функции:

- NamedPipe::CreatePipe() - создает файл в файловой системе.Обертка над mkfifo() с правами доступа 0666.
- NamedPipe::OpenPipe(int mode) - открывает канал с заданным режимом доступа.
- NamedPipe::Send(const void *buffer, size_t size) / NamedPipe::Receive(void *buffer, size_t size) -обертки над системными вызовами write и read для передачи структур Packet.

game_logic.h/cpp — реализация игры «Морской бой» Основные функции:

- void GameBoard::placeShipsRandomly() - автоматически размещает корабли (размером от 1 до 4 клеток) случайным образом, соблюдая правила расстановки и границы поля.
- Result GameBoard::processShot(int r, int c) - обрабатывает выстрел по координатам, обновляет состояние клеток (HIT/MISS) и проверяет условие окончания игры.

- `static char GameBoard::cellToChar(CellState state, bool showShips) - генерирует символьное представление клетки. Параметр showShips позволяет скрывать вражеские корабли`

`server_app.h/cpp` — центральный сервер, управляющий подключениями и сессиями. Основные функции:

- `void ServerApp::run() - основной цикл сервера, читающий сообщения из главного канала и распределяющий их по методам-обработчикам.`
- `void ServerApp::handleLogin(Packet &pkt) - регистрирует нового игрока, добавляя его в вектор players, или загружает существующего.`
- `void ServerApp::handleCreateGame(Packet &pkt)) / handleJoinGame(Packet &pkt)) - обрабатывает процесс создания игровых сессий: создает матч, связывает пару игроков и инициализирует начало боя.`

`server_app.h/cpp` — реализует логику игрового клиента и интерфейс пользователя. Основные функции:

- `void ClientApp::start(): инициализирует игрока, отправляет запрос на авторизацию и запускает основной цикл меню (создание игры, выстрел, статистика).`
- `void ClientApp::listenLoop(): фоновый метод, работающий в отдельном потоке. Ожидает пакеты от сервера и мгновенно обновляет состояние локальных полей при получении данных о ходах противника.`
- `void ClientApp::displayBoards(): визуализирует в консоли два игровых поля (свое с кораблями и поле противника с отметками выстрелов).`
- `void ClientApp::sendPacket(Packet &pkt): кратковременно открывает главный канал сервера для отправки команды и закрывает его после передачи данных.`

`main_server.cpp / main_client.cpp` — точки входа.

Результаты

В ходе курсовой работы был спроектирован и реализован программный прототип многопользовательской игры «Морской бой», использующий клиент-серверную архитектуру. Взаимодействие процессов: Успешно организована передача структурированных данных (пакетов struct Packet) между независимыми процессами сервера и нескольких клиентов через именованные каналы (Named Pipes/FIFO). Сервер корректно обрабатывает запросы от нескольких пользователей, используя мультиплексирование сообщений через единый входной канал (/tmp/sb_server_in), в то время как клиенты асинхронно получают ответы через персональные каналы. Игровая логика: Реализована полноценная логика игры: регистрация пользователей, механизм создания и поиска игровых сессий (CREATE_GAME/JOIN_GAME), автоматическая генерация игровых полей, поочередная стрельба и проверка попаданий. Сервер выступает в роли арбитра, корректно управляя состоянием игры и блокируя попытки повторных выстрелов в уже пораженные клетки (RES_REPEAT).

Синхронизация и многопоточность: В клиентском приложении успешно применена многопоточность с использованием библиотеки pthread. Фоновый поток (listenerThread) обеспечивает непрерывное чтение сообщений от сервера для обновления игрового поля в реальном времени, не блокируя основной поток ввода команд пользователем. На сервере обеспечена потокобезопасность списков активных игр и игроков с помощью мьютексов (pthread_mutex).

Пользовательский интерфейс: Реализована система визуализации игрового поля в консоли. Игроки получают актуальное отображение своего поля и поля соперника(как в реальной жизни)со скрытыми координатами его кораблей до момента попадания, что обеспечивает честный игровой процесс.

Обработка исключительных ситуаций и статистика: Программа корректно обрабатывает выход игроков (команда QUIT) и системные сигналы (SIGINT). Реализована система сохранения статистики побед и поражений в файл stats.txt, данные из которого автоматически подгружаются при каждом запуске сервера. Ресурсы системы (файлы каналов в /tmp) корректно освобождаются при завершении работы приложений.

Выводы

В ходе выполнения работы реализована полноценная игра «Морской бой», работающая на операционной системе Linux. Организовано межпроцессное взаимодействие (IPC): Связь между независимыми процессами настроена через именованные каналы (FIFO). Реализована схема взаимодействия, при которой сервер принимает запросы через общий канал, а отправляет ответы через персональные каналы клиентов. В клиентском приложении с помощью библиотеки pthread реализовано разделение задач. Это позволило одновременно обрабатывать ввод данных пользователем и в фоновом режиме принимать обновления игрового поля от сервера. Сервер спроектирован как центральный узел, который хранит состояния полей, проверяет координаты выстрелов и определяет победителя. Такой подход гарантирует честность игры и сохранность игровых данных. Для предотвращения конфликтов при одновременном доступе нескольких игроков к общим ресурсам сервера использованы мьютексы. Это обеспечивает стабильную работу системы под нагрузкой. Реализован механизм сохранения статистики в файл stats.txt. Данные о победах и поражениях автоматически загружаются при старте и обновляются по завершении матчей.

Исходная программа

```
#pragma once

#include "common.h"
#include "pipes.h"

#include <pthread.h>
#include <string>

class ClientApp {
public:
  ClientApp();
  ~ClientApp() = default;
  void start();

private:
  std::string login;
  std::string Friend;
  bool isRunning;
  pthread_t listenerThread;
  Board myBoard;
  Board friendBoard;

  static void *listenThreadWrapper(void *ctx);
  void listenLoop();

  void sendPacket(Packet &pkt);
  void displayBoards();
};
```

Листинг 1: include/client_app.h

```
#pragma once

#include <iostream>
#include <string>
#include <cstring>

#define BOARD_SIZE 10
#define MAX_LOGIN 64
#define STATS_FILE "stats.txt"

#define SERVER_MAIN_FIFO "/tmp/sb_server_in"
#define CLIENT_FIFO_PREFIX "/tmp/sb_client_"

enum CellState { EMPTY = 0, SHIP = 1, HIT = 2, MISS = 3 };
enum Result { RES_MISS, RES_HIT, RES_SUNK, RES_REPEAT, RES_LOSE };

enum class RequestType {
    LOGIN,
    CREATE_GAME,
    JOIN_GAME,
    GET_STATS,
    PLACE_SHIPS,
    SHOT,
    QUIT
```

```
};

struct Board {
    CellState cells[BOARD_SIZE][BOARD_SIZE];
};

struct Player {
    char login[MAX_LOGIN];
    Board board;
    int wins;
    int losses;
};

struct GameSession {
    char name[MAX_LOGIN];
    char player1[MAX_LOGIN];
    char player2[MAX_LOGIN];
    bool isFull;
};

struct Packet {
    RequestType cmd;
    char login[MAX_LOGIN];
    char target[MAX_LOGIN];
    int row;
    int col;
    bool is_hit;
    Result res;
    Board board;
    bool your_turn;
};
```

Листинг 2: include/common.h

```
#pragma once

#include <cstdlib>
#include <ctime>
#include <vector>
#include "common.h"

class GameBoard {
public:
  GameBoard();

  void placeShipsRandomly();

  Result processShot(int x, int y);
  Board getRawBoard() const;
  static char cellToChar(CellState state, bool showShips);
private:
  CellState cells[BOARD_SIZE][BOARD_SIZE];
  int shipsAlive;
};
```

Листинг 3: include/game_logic.h

```
#pragma once

#include "common.h"
#include "pipes.h"

#include <pthread.h>
#include <string>
#include <vector>

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

private:
  std::vector<Player> players;
  std::vector<GameSession> activeGames;
  pthread_mutex_t list_mutex;
  NamedPipe serverPipe;
  bool isRunning;

  Player *findPlayer(const std::string &login);
  void sendToClient(const std::string &login, Packet &pkt);
  void loadStatusFromFile();
  void saveStatsToFile();
  void handleCreateGame(Packet &pkt);
  void handleJoinGame(Packet &pkt);
  void handleLogin(Packet &pkt);
  void handlePlaceShips(Packet &pkt);
  void hanldeShoot(Packet &pkt);
  void handleLogout(Packet &pkt);
  void handleGetStats(Packet &pkt);
};
```

Листинг 4: include/server_app.h

```
#pragma once

#include <cerrno>
#include <cstring>
#include <fcntl.h>
#include <iostream>
#include <string>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

class NamedPipe {
public:
  std::string path;
  int fd;

  NamedPipe(std::string _path) : path(_path), fd(-1) {}

  bool CreatePipe() {
    if (mkfifo(path.c_str(), 0666) == -1) {
```

```
      if (errno != EEXIST) {
        return false;
      }
    }
    return true;
  }

  bool OpenPipe(int mode) {
    fd = open(path.c_str(), mode);
    return (fd != -1);
  }

  void ClosePipe() {
    if (fd != -1) {
      close(fd);
      fd = -1;
    }
  }

  bool Send(const void *buffer, size_t size) {
    if (fd == -1) {
      return false;
    }
    return write(fd, buffer, size) == (ssize_t)size;
  }

  bool Receive(void *buffer, size_t size) {
    if (fd == -1) {
      return false;
    }
    return read(fd, buffer, size) == (ssize_t)size;
  }
  ~NamedPipe() {
    ClosePipe();
    }
};
```

Листинг 5: include/pipes.h

```
#include "client_app.h"
#include "game_logic.h"
#include <unistd.h>
#include <ctime>

ClientApp::ClientApp() : isRunning(true) {
    memset(&myBoard, 0, sizeof(Board));
    memset(&friendBoard, 0, sizeof(Board));
}

void* ClientApp::listenThreadWrapper(void* ctx) {
    ((ClientApp*)ctx)->listenLoop();
    return nullptr;
}

void ClientApp::sendPacket(Packet &pkt) {
    NamedPipe sp(SERVER_MAIN_FIFO);
    if (sp.OpenPipe(O_WRONLY)) {
```

```
19          sp.Send(&pkt, sizeof(Packet));
20          usleep(10000);
21          sp.ClosePipe();
22      }
23  }
24
25  void ClientApp::displayBoards() {
26      std::cout << "\n YOUR BOARD FRIEND " << std::endl;
27      for (int i = 0; i < BOARD_SIZE; i++) {
28          std::cout << i << " | ";
29          for (int j = 0; j < BOARD_SIZE; j++) {
30              std::cout << GameBoard::cellToChar(myBoard.cells[i][j], true) << " ";
31          }
32          std::cout << " " << i << " | ";
33          for (int j = 0; j < BOARD_SIZE; j++) {
34              std::cout << GameBoard::cellToChar(friendBoard.cells[i][j], false) << " ";
35          }
36          std::cout << std::endl;
37      }
38  }
39
40  void ClientApp::start() {
41      srand(time(NULL));
42      std::cout << "Your login: ";
43      std::cin >> login;
44      Packet p;
45      memset(&p, 0, sizeof(p));
46      p.cmd = RequestType::LOGIN;
47      strncpy(p.login, login.c_str(), MAX_LOGIN);
48      sendPacket(p);
49
50      pthread_create(&listenerThread, NULL, listenThreadWrapper, this);
51      std::cout << "1. Create Game" << std::endl;
52      std::cout << "2. Join Game" << std::endl;
53      int mode;
54      std::cin >> mode;
55      if (mode == 1) {
56          std::cout << "Enter game name: ";
57          std::string gName;
58          std::cin >> gName;
59          p.cmd = RequestType::CREATE_GAME;
60          strncpy(p.target, gName.c_str(), MAX_LOGIN);
61          sendPacket(p);
62          std::cout << "Waiting your friend" << std::endl;
63      } else {
64          std::cout << "Enter game name to join: ";
65          std::string gName;
66          std::cin >> gName;
67          p.cmd = RequestType::JOIN_GAME;
68          strncpy(p.target, gName.c_str(), MAX_LOGIN);
69          sendPacket(p);
70      }
71      while(Friend.empty() && isRunning) {
72          usleep(100000);
73      }
74      GameBoard logic;
75      logic.placeShipsRandomly();
76      myBoard = logic.getRawBoard();
```

```
77      p.cmd = RequestType::PLACE_SHIPS;
78      p.board = myBoard;
79      sendPacket(p);
80      while (isRunning) {
81          std::cout << "MENU" << std::endl;
82          if (!Friend.empty()) {
83              std::cout << "Game with " << Friend << std::endl;
84          }
85          std::cout << "1. Show board\n2. Make a shot\n3. Exit\n4. Statistics\n> ";
86          int c;
87          std::cin >> c;
88          if (c == 1) {
89              displayBoards();
90          }
91          else if (c == 2) {
92              Packet s; memset(&s, 0, sizeof(s));
93              s.cmd = RequestType::SHOT;
94              strncpy(s.login, login.c_str(), MAX_LOGIN);
95              if (Friend.empty()) {
96                  std::cout << "Enter name your friend: ";
97                  std::cin >> Friend;
98              }
99              strncpy(s.target, Friend.c_str(), MAX_LOGIN);
100             std::cout << "Coordinates (row column): ";
101             std::cin >> s.row >> s.col;
102             sendPacket(s);
103         }
104         else if (c == 3) {
105             isRunning = false;
106             p.cmd = RequestType::QUIT;
107             sendPacket(p);
108         }
109         else if (c == 4) {
110             Packet s; memset(&s, 0, sizeof(s));
111             s.cmd = RequestType::GET_STATS;
112             strncpy(s.login, login.c_str(), MAX_LOGIN);
113             sendPacket(s);
114         }
115     }
116     pthread_join(listenerThread, NULL);
117 }
118
119 void ClientApp::listenLoop() {
120     NamedPipe mp(CLIENT_FIFO_PREFIX + login);
121     mp.CreatePipe();
122     while (isRunning) {
123         if (mp.OpenPipe(O_RDONLY)) {
124             Packet pkt;
125             if (mp.Receive(&pkt, sizeof(Packet))) {
126                 if (pkt.cmd == RequestType::JOIN_GAME) {
127                     Friend = pkt.login;
128                     std::cout << "\nGame start. Your friend: " << Friend << std::endl;
129                 } else if (pkt.cmd == RequestType::SHOT) {
130                     if (pkt.res == RES_REPEAT) {
131                         if (std::string(pkt.login) == login) {
132                             std::cout << "\n[!] You already shot here!" << std::endl;
133                         }
134                         continue;
```

```
                    }
                    if (std::string(pkt.target) == login) {
                        myBoard.cells[pkt.row][pkt.col] = pkt.is_hit ? HIT : MISS;
                    } else {
                        friendBoard.cells[pkt.row][pkt.col] = pkt.is_hit ? HIT : MISS;
                    }

                    displayBoards();
                    bool iAmShooter = (std::string(pkt.login) == login);
                    if (iAmShooter) {
                        if (pkt.res == RES_HIT) {
                            std::cout << "HIT! KEEP SHOOTING." << std::endl;
                        } else if (pkt.res == RES_SUNK) {
                            std::cout << "KILLED! KEEP SHOOTING." << std::endl;
                        } else if (pkt.res == RES_MISS) {
                            std::cout << "MISTAKE! YOUR FRIEND IS SHOOTING." << std::endl
                                ;
                        }
                    } else {
                        if (pkt.is_hit) {
                            std::cout << "GOT HIT ON YOU! FRIEND SHOOTS AGAIN." << std::
                                endl;
                        } else {
                            std::cout << "FRIEND MISSED! YOUR SHOT." << std::endl;
                        }
                    }
                    if (pkt.res == RES_LOSE) {
                        std::string winner = pkt.your_turn ? login : Friend;
                        std::cout << "GAME OVER! WINNER:" << winner << std::endl;
                        Friend = "";
                    }
                }
                else if (pkt.cmd == RequestType::GET_STATS) {
                    std::cout << "\nYOUR STATISTICS" << std::endl;
                    std::cout << "Player: " << pkt.login << std::endl;
                    std::cout << "Winners: " << pkt.row << std::endl;
                    std::cout << "Looses: " << pkt.col << std::endl;
                }
            }
            mp.ClosePipe();
        }
    }
}
```

Листинг 6: src/client/client_app.cpp

```
#include "client_app.h"
#include <iostream>


int main() {
    std::cout << "Player NAVAL BATTLE" << std::endl;
    try {
        ClientApp app;
        app.start();
    } catch (const std::exception& e) {
        std::cerr << "error player: " << e.what() << std::endl;
```

```
        return 1;
    }
    return 0;
}
```

Листинг 7: src/client/main_client.cpp

```
#include "server_app.h"
#include <iostream>
#include <csignal>

ServerApp* globalAppPtr = nullptr;
void signalHandler(int signum) {
    if (globalAppPtr) {
        delete globalAppPtr;
        globalAppPtr = nullptr;
    }
    exit(signum);
}

int main() {
    signal(SIGINT, signalHandler);
    try {
        ServerApp app;
        app.run();
    } catch (const std::exception& e) {
        std::cerr << "error server: " << e.what() << std::endl;
        return 1;
    }
    return 0;
}
```

Листинг 8: src/server/main_server.cpp

```
#include "server_app.h"
#include <algorithm>
#include <fstream>
#include <iostream>
#include <cstring>

ServerApp::ServerApp() : serverPipe(SERVER_MAIN_FIFO), isRunning(true) {
    pthread_mutex_init(&list_mutex, NULL);
    serverPipe.CreatePipe();
    loadStatusFromFile();
}

void ServerApp::handleCreateGame(Packet &pkt) {
    pthread_mutex_lock(&list_mutex);
    GameSession newGame;
    strncpy(newGame.name, pkt.target, MAX_LOGIN);
    strncpy(newGame.player1, pkt.login, MAX_LOGIN);
    newGame.player2[0] = '\0';
    newGame.isFull = false;
    activeGames.push_back(newGame);
    pthread_mutex_unlock(&list_mutex);
```

```
22      std::cout << "(server)Game created: " << pkt.target << " by " << pkt.login << std
            ::endl;
23  }
24
25  void ServerApp::handleJoinGame(Packet &pkt) {
26      pthread_mutex_lock(&list_mutex);
27      for (auto &game : activeGames) {
28          if (std::string(game.name) == pkt.target && !game.isFull) {
29              strncpy(game.player2, pkt.login, MAX_LOGIN);
30              game.isFull = true;
31              Packet confirm;
32              confirm.cmd = RequestType::JOIN_GAME;
33              strncpy(confirm.login, game.player2, MAX_LOGIN);
34              sendToClient(game.player1, confirm);
35              strncpy(confirm.login, game.player1, MAX_LOGIN);
36              sendToClient(game.player2, confirm);
37              break;
38          }
39      }
40      pthread_mutex_unlock(&list_mutex);
41  }
42
43  void ServerApp::loadStatusFromFile() {
44      std::ifstream ifs(STATS_FILE);
45      if (!ifs.is_open()) {
46          return;
47      }
48      std::string name;
49      int w;
50      int l;
51      while (ifs >> name >> w >> l) {
52          Player p;
53          memset(&p, 0, sizeof(Player));
54          strncpy(p.login, name.c_str(), MAX_LOGIN);
55          p.wins = w;
56          p.losses = l;
57          players.push_back(p);
58      }
59      ifs.close();
60      std::cout << "(server)Loaded stats for " << players.size() << " players." << std::
            endl;
61  }
62
63  Player* ServerApp::findPlayer(const std::string &login) {
64      for (auto &p : players) {
65          if (std::string(p.login) == login) {
66              return &p;
67          }
68      }
69      return nullptr;
70  }
71
72  void ServerApp::sendToClient(const std::string &login, Packet &pkt) {
73      std::string clientPath = CLIENT_FIFO_PREFIX + login;
74      NamedPipe clientPipe(clientPath);
75      if (clientPipe.OpenPipe(O_WRONLY)) {
76          clientPipe.Send(&pkt, sizeof(Packet));
77          clientPipe.ClosePipe();
```

```
78      }
79  }
80
81  void ServerApp::saveStatsToFile() {
82      std::ofstream ofs(STATS_FILE);
83      if (!ofs.is_open()) {
84          return;
85      }
86      for (const auto& p : players) {
87          ofs << p.login << " " << p.wins << " " << p.losses << std::endl;
88      }
89      ofs.close();
90  }
91
92  void ServerApp::handleLogin(Packet &pkt) {
93      pthread_mutex_lock(&list_mutex);
94      Player* p = findPlayer(pkt.login);
95      if (!p) {
96          Player newPlayer;
97          memset(&newPlayer, 0, sizeof(Player));
98          strncpy(newPlayer.login, pkt.login, MAX_LOGIN);
99          players.push_back(newPlayer);
100         std::cout << "(login)New player: " << pkt.login << std::endl;
101     } else {
102         std::cout << "(login))Player returned: " << pkt.login << std::endl;
103     }
104     pthread_mutex_unlock(&list_mutex);
105 }
106
107 void ServerApp::handlePlaceShips(Packet &pkt) {
108     pthread_mutex_lock(&list_mutex);
109     Player* p = findPlayer(pkt.login);
110     if (p) {
111         p->board = pkt.board;
112         std::cout << "[BOARD] " << pkt.login << " placed ships.\n";
113     }
114     pthread_mutex_unlock(&list_mutex);
115 }
116
117 void ServerApp::hanldeShoot(Packet &pkt) {
118     pthread_mutex_lock(&list_mutex);
119     Player* shooter = findPlayer(pkt.login);
120     Player* victim = findPlayer(pkt.target);
121     if (shooter && victim) {
122         CellState &cell = victim->board.cells[pkt.row][pkt.col];
123         if (cell == HIT || cell == MISS) {
124             pkt.res = RES_REPEAT;
125             pkt.is_hit = false;
126             pkt.your_turn = true;
127         } else if (cell == SHIP) {
128             cell = HIT;
129             pkt.is_hit = true;
130             pkt.your_turn = true;
131             bool any_ships_left = false;
132             for(int i=0; i<BOARD_SIZE; i++) {
133                 for(int j=0; j<BOARD_SIZE; j++){
134                     if(victim->board.cells[i][j] == SHIP) {
135                         any_ships_left = true;
```

```
                    }
                }
            }
            if (!any_ships_left) {
                pkt.res = RES_LOSE;
                shooter->wins++;
                victim->losses++;
                saveStatsToFile();
            } else {
                bool part_of_ship_alive = false;
                int dr[] = {-1, 1, 0, 0};
                int dc[] = {0, 0, -1, 1};
                for(int i = 0; i < 4; i++) {
                    int nr = pkt.row + dr[i];
                    int nc = pkt.col + dc[i];
                    if(nr >= 0 && nr < BOARD_SIZE && nc >= 0 && nc < BOARD_SIZE) {
                        if(victim->board.cells[nr][nc] == SHIP) {
                            part_of_ship_alive = true;
                            break;
                        }
                    }
                }
                if (part_of_ship_alive) {
                    pkt.res = RES_HIT;
                } else {
                    pkt.res = RES_SUNK;
                }
            }
        } else {
            cell = MISS;
            pkt.is_hit = false;
            pkt.res = RES_MISS;
            pkt.your_turn = false;
        }
        sendToClient(pkt.login, pkt);
        Packet notifyVictim = pkt;
        notifyVictim.your_turn = !pkt.your_turn;
        sendToClient(pkt.target, notifyVictim);
    }
    pthread_mutex_unlock(&list_mutex);
}

void ServerApp::handleLogout(Packet &pkt) {
    pthread_mutex_lock(&list_mutex);
    std::cout << "(quit)Player disconnected: " << pkt.login << std::endl;
    pthread_mutex_unlock(&list_mutex);
}

void ServerApp::run() {
    std::cout << "(server)Waiting for players..." << std::endl;
    while (isRunning) {
        if (serverPipe.OpenPipe(O_RDONLY)) {
            Packet pkt;
            while (serverPipe.Receive(&pkt, sizeof(Packet))) {
                switch (pkt.cmd) {
                    case RequestType::LOGIN: handleLogin(pkt); break;
                    case RequestType::PLACE_SHIPS: handlePlaceShips(pkt); break;
                    case RequestType::SHOT: hanldeShoot(pkt); break;
```

```
                    case RequestType::QUIT: handleLogout(pkt); break;
                    case RequestType::GET_STATS: handleGetStats(pkt); break;
                    case RequestType::CREATE_GAME: handleCreateGame(pkt); break;
                    case RequestType::JOIN_GAME: handleJoinGame(pkt); break;
                }
            }
            serverPipe.ClosePipe();
        }
    }
}

void ServerApp::handleGetStats(Packet &pkt) {
    pthread_mutex_lock(&list_mutex);
    Player* p = findPlayer(pkt.login);
    if (p) {
        pkt.row = p->wins;
        pkt.col = p->losses;
        std::cout << "(stats)Sending stats to " << pkt.login << ": W:" << p->wins << "
            L:" << p->losses << std::endl;
    }
    pthread_mutex_unlock(&list_mutex);
    sendToClient(pkt.login, pkt);
}

ServerApp::~ServerApp() {
    saveStatsToFile();
    pthread_mutex_destroy(&list_mutex);
    unlink(SERVER_MAIN_FIFO);
}
```

Листинг 9: src/server/server_app.cpp

```
#include "game_logic.h"
#include <algorithm>

GameBoard::GameBoard() {
    for (int i = 0; i < BOARD_SIZE; i++) {
        for (int j = 0; j < BOARD_SIZE; j++) {
            cells[i][j] = EMPTY;
        }
    }
    shipsAlive = 20;
}

void GameBoard::placeShipsRandomly() {
    for (int i = 0; i < BOARD_SIZE; i++) {
        for (int j = 0; j < BOARD_SIZE; j++) {
            cells[i][j] = EMPTY;
        }
    }
    int shipSizes[] = {4, 3, 3, 2, 2, 2, 1, 1, 1, 1};
    for (int size : shipSizes) {
        bool placed = false;
        while (!placed) {
            int r = rand() % BOARD_SIZE;
            int c = rand() % BOARD_SIZE;
            int dir = rand() % 2;
```

```
            bool canPlace = true;
            for (int i = 0; i < size; i++) {
                int currR = r + (dir == 1 ? i : 0);
                int currC = c + (dir == 0 ? i : 0);
                if (currR >= BOARD_SIZE || currC >= BOARD_SIZE || cells[currR][currC]
                    != EMPTY) {
                    canPlace = false;
                    break;
                }
                for (int dr = -1; dr <= 1; dr++) {
                    for (int dc = -1; dc <= 1; dc++) {
                        int nr = currR + dr;
                        int nc = currC + dc;
                        if (nr >= 0 && nr < BOARD_SIZE && nc >= 0 && nc < BOARD_SIZE) {
                            if (cells[nr][nc] == SHIP) {
                                canPlace = false;
                            }
                        }
                    }
                }
            }
            if (canPlace) {
                for (int i = 0; i < size; i++) {
                    cells[r + (dir == 1 ? i : 0)][c + (dir == 0 ? i : 0)] = SHIP;
                }
                placed = true;
            }
        }
    }
}

Result GameBoard::processShot(int r, int c) {
    if (r < 0 || r >= BOARD_SIZE || c < 0 || c >= BOARD_SIZE) {
        return RES_REPEAT;
    }
    if (cells[r][c] == SHIP) {
        cells[r][c] = HIT;
        shipsAlive--;
        if (shipsAlive == 0) {
            return RES_LOSE;
        }
        return RES_HIT;
    } else if (cells[r][c] == EMPTY) {
        cells[r][c] = MISS;
        return RES_MISS;
    }
    return RES_REPEAT;
}

Board GameBoard::getRawBoard() const {
    Board b;
    for (int i = 0; i < BOARD_SIZE; i++) {
        for (int j = 0; j < BOARD_SIZE; j++) {
            b.cells[i][j] = cells[i][j];
        }
    }
    return b;
}
```

```
83
84  char GameBoard::cellToChar(CellState state, bool showShips) {
85      switch (state) {
86          case SHIP: return showShips ? 'K' : ' ';
87          case HIT: return 'X';
88          case MISS: return '*';
89          case EMPTY: return '.';
90          default: return ' ';
91      }
92  }
```

Листинг 10: src/game_logic.cpp

Strace сервер

```
execve("./server", ["./server"], 0x7ffd1c1d2040 /* 26 vars */) = 0
brk(NULL)                               = 0x58e525296000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffef1508040) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
↪  0x70dca9028000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=29000, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 29000, PROT_READ, MAP_PRIVATE, 3, 0) = 0x70dca9020000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2260296, ...}, AT_EMPTY_PATH) =
↪  0
mmap(NULL, 2275520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70dca8c00000
mprotect(0x70dca8c9a000, 1576960, PROT_NONE) = 0
mmap(0x70dca8c9a000, 1118208, PROT_READ|PROT_EXEC,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9a000) = 0x70dca8c9a000
mmap(0x70dca8dab000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
↪  0x1ab000) = 0x70dca8dab000
mmap(0x70dca8e1b000, 57344, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x21a000) = 0x70dca8e1b000
mmap(0x70dca8e29000, 10432, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x70dca8e29000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...}, AT_EMPTY_PATH) =
↪  0
mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70dca9000000
mmap(0x70dca9003000, 94208, PROT_READ|PROT_EXEC,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x70dca9003000
mmap(0x70dca901a000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
↪  0x1a000) = 0x70dca901a000
mmap(0x70dca901e000, 8192, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d000) = 0x70dca901e000
```

```
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\0P\237\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
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"...,
↪  48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32\230\┘
↪  266\235"..., 68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) =
↪  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, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70dca8800000
mprotect(0x70dca8828000, 2023424, PROT_NONE) = 0
mmap(0x70dca8828000, 1658880, PROT_READ|PROT_EXEC,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x70dca8828000
mmap(0x70dca89bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
↪  0x1bd000) = 0x70dca89bd000
mmap(0x70dca8a16000, 24576, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x70dca8a16000
mmap(0x70dca8a1c000, 52816, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x70dca8a1c000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) =
↪  0
mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70dca8f19000
mmap(0x70dca8f27000, 507904, PROT_READ|PROT_EXEC,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x70dca8f27000
mmap(0x70dca8fa3000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
↪  0x8a000) = 0x70dca8fa3000
mmap(0x70dca8ffe000, 8192, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x70dca8ffe000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
↪  0x70dca8f17000
arch_prctl(ARCH_SET_FS, 0x70dca8f183c0) = 0
set_tid_address(0x70dca8f18690)         = 9278
set_robust_list(0x70dca8f186a0, 24)     = 0
rseq(0x70dca8f18d60, 0x20, 0, 0x53053053) = 0
mprotect(0x70dca8a16000, 16384, PROT_READ) = 0
mprotect(0x70dca8ffe000, 4096, PROT_READ) = 0
mprotect(0x70dca901e000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
↪  0x70dca8f15000
mprotect(0x70dca8e1b000, 45056, PROT_READ) = 0
mprotect(0x58e4ef4b3000, 4096, PROT_READ) = 0
mprotect(0x70dca9062000, 8192, PROT_READ) = 0
```

```
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
↪  rlim_max=RLIM64_INFINITY}) = 0
munmap(0x70dca9020000, 29000)           = 0
getrandom("\xd5\x86\x3f\x49\xc3\x5c\xf6\x62", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x58e525296000
brk(0x58e5252b7000)                     = 0x58e5252b7000
futex(0x70dca8e2977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
rt_sigaction(SIGINT, {sa_handler=0x58e4ef4aaa09, sa_mask=[INT],
↪  sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x70dca8842520},
↪  {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
mknodat(AT_FDCWD, "/tmp/sb_server_in", S_IFIFO|0666) = 0
openat(AT_FDCWD, "stats.txt", O_RDONLY) = 3
read(3, "V 1 0\nM 0 1\nm 0 1\nv 1 0\n", 8191) = 24
read(3, "", 8191)                       = 0
close(3)                                = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...},
↪  AT_EMPTY_PATH) = 0
write(1, "(server)Loaded stats for 4 playe"..., 36) = 36
write(1, "(server)Waiting for players...\n", 31) = 31
openat(AT_FDCWD, "/tmp/sb_server_in", O_RDONLY) = 3
read(3, "\0\0\0\0kiko\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 552)
↪  = 552
write(1, "(login)New player: kiko\n", 24) = 24
read(3, "", 552)                        = 0
close(3)                                = 0
openat(AT_FDCWD, "/tmp/sb_server_in", O_RDONLY) = 3
read(3, "\1\0\0\0kiko\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 552)
↪  = 552
write(1, "(server)Game created: fifa by ki"..., 35) = 35
read(3, "", 552)                        = 0
close(3)                                = 0
openat(AT_FDCWD, "/tmp/sb_server_in", O_RDONLY) = 3
read(3, "\0\0\0\0er\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"...,
↪  552) = 552
write(1, "(login)New player: er\n", 22) = 22
read(3, "", 552)                        = 0
close(3)                                = 0
openat(AT_FDCWD, "/tmp/sb_server_in", O_RDONLY) = 3
read(3, "\2\0\0\0er\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"...,
↪  552) = 552
openat(AT_FDCWD, "/tmp/sb_client_kiko", O_WRONLY) = 4
write(4, "\2\0\0\0er\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"...,
↪  552) = 552
close(4)                                = 0
openat(AT_FDCWD, "/tmp/sb_client_er", O_WRONLY) = 4
write(4, "\2\0\0\0kiko\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"...,
↪  552) = 552
close(4)                                = 0
read(3, "", 552)                        = 0
close(3)                                = 0
openat(AT_FDCWD, "/tmp/sb_server_in", O_RDONLY) = 3
```

```
read(3, "\4\0\0\0er\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"...,
↪  552) = 552
write(1, "[BOARD] er placed ships.\n", 25) = 25
read(3, "", 552)                        = 0
close(3)                                = 0
openat(AT_FDCWD, "/tmp/sb_server_in", O_RDONLY) = 3
read(3, "\4\0\0\0kiko\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 552)
↪  = 552
write(1, "[BOARD] kiko placed ships.\n", 27) = 27
read(3, "", 552)                        = 0
close(3)                                = 0
openat(AT_FDCWD, "/tmp/sb_server_in", O_RDONLY) = 3
read(3, "\5\0\0\0kiko\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 552)
↪  = 552
openat(AT_FDCWD, "/tmp/sb_client_kiko", O_WRONLY) = 4
write(4, "\5\0\0\0kiko\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"...,
↪  552) = 552
close(4)                                = 0
openat(AT_FDCWD, "/tmp/sb_client_er", O_WRONLY) = 4
write(4, "\5\0\0\0kiko\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"...,
↪  552) = 552
close(4)                                = 0
read(3, "", 552)                        = 0
close(3)                                = 0
openat(AT_FDCWD, "/tmp/sb_server_in", O_RDONLY) = ? ERESTARTSYS (To be
↪  restarted if SA_RESTART is set)
--- SIGINT {si_signo=SIGINT, si_code=SI_KERNEL} ---
exit_group(2)                           = ?
+++ exited with 2 +++
```

Strace клиент

```
execve("./client", ["./client"], 0x7fff2ba4fd80 /* 26 vars */) = 0
brk(NULL)                               = 0x5eb679340000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd4e007630) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
↪  0x79b9e380d000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=29000, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 29000, PROT_READ, MAP_PRIVATE, 3, 0) = 0x79b9e3805000
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\0"...,
↪  832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2260296, ...}, AT_EMPTY_PATH) =
↪  0
mmap(NULL, 2275520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x79b9e3400000
mprotect(0x79b9e349a000, 1576960, PROT_NONE) = 0
mmap(0x79b9e349a000, 1118208, PROT_READ|PROT_EXEC,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9a000) = 0x79b9e349a000
mmap(0x79b9e35ab000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
↪  0x1ab000) = 0x79b9e35ab000
```

```
mmap(0x79b9e361b000, 57344, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x21a000) = 0x79b9e361b000
mmap(0x79b9e3629000, 10432, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x79b9e3629000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...}, AT_EMPTY_PATH) =
↪  0
mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x79b9e37e5000
mmap(0x79b9e37e8000, 94208, PROT_READ|PROT_EXEC,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x79b9e37e8000
mmap(0x79b9e37ff000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
↪  0x1a000) = 0x79b9e37ff000
mmap(0x79b9e3803000, 8192, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d000) = 0x79b9e3803000
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\0P\237\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
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"...,
↪  48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32\230\」
↪  266\235"..., 68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) =
↪  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, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x79b9e3000000
mprotect(0x79b9e3028000, 2023424, PROT_NONE) = 0
mmap(0x79b9e3028000, 1658880, PROT_READ|PROT_EXEC,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x79b9e3028000
mmap(0x79b9e31bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
↪  0x1bd000) = 0x79b9e31bd000
mmap(0x79b9e3216000, 24576, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x79b9e3216000
mmap(0x79b9e321c000, 52816, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x79b9e321c000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) =
↪  0
mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x79b9e36fe000
mmap(0x79b9e370c000, 507904, PROT_READ|PROT_EXEC,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x79b9e370c000
mmap(0x79b9e3788000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
↪  0x8a000) = 0x79b9e3788000
```

```
mmap(0x79b9e37e3000, 8192, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x79b9e37e3000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
↪  0x79b9e36fc000
arch_prctl(ARCH_SET_FS, 0x79b9e36fd3c0) = 0
set_tid_address(0x79b9e36fd690)         = 9284
set_robust_list(0x79b9e36fd6a0, 24)     = 0
rseq(0x79b9e36fdd60, 0x20, 0, 0x53053053) = 0
mprotect(0x79b9e3216000, 16384, PROT_READ) = 0
mprotect(0x79b9e37e3000, 4096, PROT_READ) = 0
mprotect(0x79b9e3803000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
↪  0x79b9e36fa000
mprotect(0x79b9e361b000, 45056, PROT_READ) = 0
mprotect(0x5eb648878000, 4096, PROT_READ) = 0
mprotect(0x79b9e3847000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
↪  rlim_max=RLIM64_INFINITY}) = 0
munmap(0x79b9e3805000, 29000)           = 0
getrandom("\x73\x5e\x6e\x9c\x86\x70\x23\x76", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5eb679340000
brk(0x5eb679361000)                     = 0x5eb679361000
futex(0x79b9e362977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...},
↪  AT_EMPTY_PATH) = 0
write(1, "Player NAVAL BATTLE\n", 20)   = 20
write(1, "Your login: ", 12)            = 12
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...},
↪  AT_EMPTY_PATH) = 0
read(0, "kiko\n", 1024)                 = 5
openat(AT_FDCWD, "/tmp/sb_server_in", O_WRONLY) = 3
write(3, "\0\0\0\0kiko\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"...,
↪  552) = 552
clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=0, tv_nsec=10000000}, NULL) = 0
close(3)                                = 0
rt_sigaction(SIGRT_1, {sa_handler=0x79b9e3091870, sa_mask=[],
↪  sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO,
↪  sa_restorer=0x79b9e3042520}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
↪  0x79b9e27ff000
mprotect(0x79b9e2800000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_S
↪  YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
↪  child_tid=0x79b9e2fff910, parent_tid=0x79b9e2fff910, exit_signal=0,
↪  stack=0x79b9e27ff000, stack_size=0x7fff00, tls=0x79b9e2fff640} =>
↪  {parent_tid=[9285]}, 88) = 9285
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
write(1, "1. Create Game\n", 15)        = 15
write(1, "2. Join Game\n", 13)          = 13
```

```
read(0, "1\n", 1024)                    = 2
write(1, "Enter game name: ", 17)       = 17
read(0, "fifa\n", 1024)                 = 5
openat(AT_FDCWD, "/tmp/sb_server_in", O_WRONLY) = 4
write(4, "\1\0\0\0kiko\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"...,
↪  552) = 552
clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=0, tv_nsec=10000000}, NULL) = 0
close(4)                                = 0
write(1, "Waiting your friend\n", 20)   = 20
clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=0, tv_nsec=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, 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=100000000}, NULL) = 0
openat(AT_FDCWD, "/tmp/sb_server_in", O_WRONLY) = 4
write(4, "\4\0\0\0kiko\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"...,
↪  552) = 552
clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=0, tv_nsec=10000000}, NULL) = 0
close(4)                                = 0
write(1, "MENU\n", 5)                    = 5
write(1, "Game with er\n", 13)           = 13
write(1, "1. Show board\n2. Make a shot\n3. "..., 51) = 51
write(1, "> ", 2)                        = 2
read(0, "1\n", 1024)                    = 2
write(1, "\n", 1)                        = 1
write(1, "    YOUR BOARD                 F"..., 39) = 39
write(1, "0 | . . K K K K . K . .     0 | "..., 53) = 53
write(1, "1 | . . . . . . . . . K     1 | "..., 53) = 53
write(1, "2 | K . . . . . . . . K     2 | "..., 53) = 53
write(1, "3 | . . . . . . . . . .     3 | "..., 53) = 53
write(1, "4 | . . . . . . . . . .     4 | "..., 53) = 53
write(1, "5 | K K . K K K . K . .     5 | "..., 53) = 53
write(1, "6 | . . . . . . . . . .     6 | "..., 53) = 53
write(1, "7 | . . K . . . . . K .     7 | "..., 53) = 53
write(1, "8 | . . K . . K . . K .     8 | "..., 53) = 53
write(1, "9 | . . . . . . . . K .     9 | "..., 53) = 53
write(1, "MENU\n", 5)                    = 5
write(1, "Game with er\n", 13)           = 13
write(1, "1. Show board\n2. Make a shot\n3. "..., 51) = 51
write(1, "> ", 2)                        = 2
read(0, "2\n", 1024)                    = 2
write(1, "Coordinates (row column): ", 26) = 26
read(0, "0 0\n", 1024)                  = 4
openat(AT_FDCWD, "/tmp/sb_server_in", O_WRONLY) = 4
write(4, "\5\0\0\0kiko\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"...,
↪  552) = 552
clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=0, tv_nsec=10000000}, NULL) = 0
close(4)                                = 0
write(1, "MENU\n", 5)                    = 5
write(1, "Game with er\n", 13)           = 13
write(1, "1. Show board\n2. Make a shot\n3. "..., 51) = 51
write(1, "> ", 2)                        = 2
read(0, 0x5eb6793522c0, 1024)           = ? ERESTARTSYS (To be restarted if
↪  SA_RESTART is set)
--- SIGINT {si_signo=SIGINT, si_code=SI_KERNEL} ---
+++ killed by SIGINT +++
```