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»

Группа: М8О-209БВ-24

Студент: Махова А.Б.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 22.12.25

Москва, 2025

Постановка задачи

Вариант 1.

Необходимо разработать консольную клиент-серверную игру «Морской бой». Программа должна состоять из двух отдельных приложений: серверного и клиентского. Сначала запускается сервер, после чего клиенты могут подключаться к нему. Сервер координирует взаимодействие между клиентами и управляет игровыми сессиями. Обмен данными между сервером и клиентами должен быть реализован с использованием 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. При запуске клиентского приложения игроку предоставляется возможность создать новую игру, указав её имя, либо присоединиться к уже существующей игре по имени, либо запросить у сервера список уже существующих игр.

Общий метод и алгоритм решения

Использованные системные вызовы:

Общие для клиента и сервера:

- mkfifo(const char *pathname, mode_t mode) — создание именованного канала для взаимодействия процессов.
- open(const char *pathname, int flags) — открытие FIFO для чтения или записи.
- close(int fd) — закрытие файлового дескриптора.
- unlink(const char *pathname) — удаление FIFO.
- read(int fd, void *buf, size_t count) — чтение данных из FIFO.
- write(int fd, const void *buf, size_t count) — запись данных в FIFO.
- pthread_create() - создание нового потока
- pthread_cancel() - отмена выполнения потока
- pthread_join() - ожидание завершения потока
- pthread_mutex_lock() - блокировка мьютекса
- pthread_mutex_unlock() - разблокировка мьютекса
- pthread_mutex_destroy() - уничтожение мьютекса
- sleep() - приостановка выполнения на указанное время

Метод решения

Для реализации консольной сетевой игры «Морской бой» выбрана архитектура клиентсервер с топологией «Звезда». Взаимодействие между процессами (IPC) организовано через именованные каналы (Named Pipes/FIFO) в операционной системе Linux.

• Сервер является центральным узлом, который хранит состояние всех подключенных пользователей, игровые поля и логику матчей. Он слушает один общий канал для входящих запросов.

• Клиенты отвечают за взаимодействие с пользователем (ввод команд, вывод игрового поля). Каждый клиент создает собственный уникальный канал для получения ответов от сервера.

• Для обеспечения асинхронного приема сообщений на клиенте используется библиотека POSIX Threads (pthread): отдельный поток отвечает за чтение данных из канала, в то время как основной поток обрабатывает пользовательский ввод:

– серверного приложения;
– клиентского приложения.

Алгоритм решения задачи:

1. Запуск Сервера:

• Сервер создает главный именованный канал (FIFO) по фиксированному пути (/tmp/battleship_server_pipe) с правами доступа 0666.

• Канал открывается в режиме O_RDWR (чтение и запись), чтобы дескриптор оставался открытым даже при отсутствии активных клиентов (предотвращение получения EOF в цикле чтения).

• Сервер входит в бесконечный цикл ожидания пакетов (struct Packet) фиксированного размера из главного канала.

2. Запуск Клиента и Авторизация:

• При запуске клиент запрашивает логин пользователя.

• Создается уникальный именованный канал для приема сообщений: /tmp/client_<login>.

• Запускается фоновый поток (listener_thread) с использованием 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.

3. Обработка запросов на Сервере:

• Сервер считывает Packet из общего канала.

• Доступ к списку игроков защищен мьютексом (pthread_mutex) для предотвращения состояний гонки.

• В зависимости от типа пакета (type) вызывается соответствующий метод обработки (handleLogin, handleInvite, handleShoot и т.д.).

• Для отправки ответа сервер временно открывает личный канал конкретного клиента (по пути /tmp/client_<target>) в режиме O_WRONLY, записывает ответ и закрывает дескриптор.

4. Организация матча (Лобби):

• Клиент вводит команду "/create <game_name>", если он не в игре.

• Отправляется пакет типа CREATE_GAME с именем игры.

• Сервер проверяет, что игрок не в игре и что игра с таким именем не существует.

• Создается новая игровая комната (GameRoom) с указанным именем, создателем становится отправитель.

• Игроку отправляется подтверждение, а также всем онлайн игрокам (не в игре) рассылается обновленный список игр.

5. Присоединение к игре

- Клиент: отправка JOIN_GAME с именем желаемой игры
- Сервер: проверка существования игры и наличия свободного места
- При успехе: добавление второго игрока, начало игры
- Рассылка уведомлений об изменении списка доступных игр

6. Подсчет и вывод статистики

- Сервер: сбор статистики (игры, победы, выстрелы, попадания)
- Клиент: запрос статистики командой /stats (GET_STATS)
- Сервер: вычисление процентов (точность, процент побед)
- Клиент: вывод форматированной статистики в консоль

7. Игровой процесс (Стрельба):

- Пользователь вводит команду /shoot X Y.
- Сервер проверяет, является ли ход игрока текущим (isTurn).
- Выполняется проверка координат на поле противника. Если попадание (RES_HIT):
  – Состояние клетки меняется на HIT.
  – Ход остается у атакующего.
  – Атакующему отправляется обновленная карта противника («Радар», где корабли скрыты).
  – Жертве отправляется карта её собственного флота с отображением повреждения.
- Если промах (RES_MISS):
  – Состояние клетки меняется на MISS.
  – Ход переходит к противнику.

8. Завершение игры:

- Если после выстрела счетчик живых клеток кораблей становится равен нулю, сервер фиксирует результат RES_LOSE.
- Обоим игрокам отправляется пакет S_GAME_OVER с соответствующим сообщением (Победа/Поражение).
- Игровые статусы сбрасываются (inGame = false), игроки возвращаются в общее "меню"и могут начинать новые партии.

9. Обработка отключений:

- При вводе команды /quit клиент отправляет пакет LOGOUT и завершает работу.

• Сервер удаляет игрока из вектора активных пользователей.

• Если игрок находился в активной партии, его противнику автоматически присуждается победа, и он возвращается в меню.

10. Управление ресурсами:

• При завершении работы программы (как клиента, так и сервера) выполняется удаление файлов каналов из файловой системы с помощью системного вызова unlink, чтобы избежать накопления мусорных файлов в директории /tmp.

Код программы

ClientApp.h

```
#pragma once

#include "protocol.h"
#include "wrappers.h"

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

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

private:
  std::string login;
  std::string currentGame;
  bool isRunning;
  bool inGame;
  pthread_t listenerThread;
```

```
  static void *listenThreadWrapper(void *context);
  void listenLoop();

  void sendPacket(Packet &pkt);
  void showMainMenu();
  void showGameMenu();
};
```

GameLogic.h

```
#pragma once

#include <cstdlib>
#include <ctime>
#include <vector>

enum CellState { EMPTY = 0, SHIP = 1, MISS = 2, HIT = 3 };

enum ShotResult { RES_MISS, RES_HIT, RES_SUNK, RES_REPEAT, RES_LOSE };

class GameBoard {
public:
  GameBoard();

  void placeShipsRandomly();

  ShotResult processShot(int x, int y);

  int getCell(int x, int y) const { return grid[y][x]; }

  void getBoardString(char *buffer, bool showShips);

private:
  static const int SIZE = 10;
```

```
  int grid[SIZE][SIZE];
  int shipsAlive;
};
```

protocol.h

```
#pragma once

#include "GameLogic.h"

#include <string>

#define SERVER_PIPE "/tmp/battleship_server_pipe"
#define CLIENT_PIPE_PREFIX "/tmp/client_"

enum MsgType {
  LOGIN,
  CREATE_GAME,
  JOIN_GAME,
  LEAVE_GAME,
  SHOOT,
  LOGOUT,
  GET_STATS,
  GET_GAME_LIST,
  S_MSG,
  S_GAME_LIST,
  S_GAME_CREATED,
  S_GAME_START,
  S_SHOT_RESULT,
  S_GAME_OVER,
  S_BOARD,
  S_STATS
};

struct Packet {
```

```
  int type;
  char sender[32];
  char gameName[64];
  char payload[512];
  int x;
  int y;
  int shotResult;
};

struct Player {
  std::string login;
  bool inGame;
  std::string gameName;
  GameBoard board;
  bool isTurn;
  std::string opponent;
};

struct GameRoom {
  std::string name;
  std::string creator;
  std::string player1;
  std::string player2;
  bool isFull;
  bool isActive;
};

struct PlayerStats {
  std::string login;
  int gamesPlayed;
  int wins;
  int losses;
  int totalShots;
  int hits;
```

```
  double accuracy;
};
```

ServerApp.h

```
#pragma once

#include "protocol.h"
#include "wrappers.h"

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

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

private:
  std::vector<Player> players;
  std::vector<GameRoom> gameRooms;
  std::map<std::string, PlayerStats> playerStats;
  pthread_mutex_t list_mutex;
  NamedPipe serverPipe;
  bool isRunning;

  Player *findPlayer(const std::string &login);
  GameRoom *findGameRoom(const std::string &gameName);
  PlayerStats *getPlayerStats(const std::string &login);
  void sendToClient(const std::string &login, Packet &pkt);
  void sendBoard(Player *pTarget, GameBoard &boardOwner, bool showShips,
                 const char *title);
```

```
  void updateStatsAfterGame(const std::string &winner, const std::string &loser);
  void sendGameList(const std::string &login);

  void handleLogin(Packet &pkt);
  void handleCreateGame(Packet &pkt);
  void handleJoinGame(Packet &pkt);
  void handleLeaveGame(Packet &pkt);
  void handleShoot(Packet &pkt);
  void handleLogout(Packet &pkt);
  void handleGetStats(Packet &pkt);
  void startGame(GameRoom &room);
};
```

wrappers.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 create() {
    if (mkfifo(path.c_str(), 0666) == -1) {
```

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

void removePipe() { unlink(path.c_str()); }

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;
}
};
```

client_main.c

```
#include "ClientApp.h"

int main() {
  ClientApp client;
  client.start();
  return 0;
}
```

server_main.cpp

```
#include "ServerApp.h"

#include <cstdlib>
#include <ctime>

int main() {
  std::srand(std::time(nullptr));
  ServerApp server;
  server.run();
  return 0;
}
```

game_logic.cpp

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>

GameBoard::GameBoard() {
  shipsAlive = 0;
  for (int i = 0; i < SIZE; ++i) {
    for (int j = 0; j < SIZE; ++j) {
      grid[i][j] = EMPTY;
```

```
    }
  }
}

void GameBoard::placeShipsRandomly() {
  static const int BATTLESHIP = 4;
  static const int CRUISER = 3;
  static const int DESTROYER = 2;
  static const int SUBMARINE = 1;

  for (int i = 0; i < SIZE; ++i) {
    for (int j = 0; j < SIZE; ++j) {
      grid[i][j] = EMPTY;
    }
  }
  shipsAlive = 0;

  int ships[] = {BATTLESHIP, CRUISER,   CRUISER,   DESTROYER, DESTROYER,
                 DESTROYER,  SUBMARINE, SUBMARINE, SUBMARINE, SUBMARINE};

  for (int len : ships) {
    bool placed = false;
    while (!placed) {
      int row = std::rand() % SIZE;
      int col = std::rand() % SIZE;
      bool horizontal = std::rand() % 2;

      bool canPlace = true;
      for (int k = 0; k < len; ++k) {
        int nrow = row + (horizontal ? 0 : k);
        int ncol = col + (horizontal ? k : 0);

        if (nrow >= SIZE || ncol >= SIZE || grid[nrow][ncol] != EMPTY) {
          canPlace = false;
```

```
          break;
        }
      }

      if (canPlace) {
        for (int k = 0; k < len; ++k) {
          int nrow = row + (horizontal ? 0 : k);
          int ncol = col + (horizontal ? k : 0);
          grid[nrow][ncol] = SHIP;
          shipsAlive++;
        }
        placed = true;
      }
    }
  }
}

ShotResult GameBoard::processShot(int x, int y) {
  if (x < 0 || x >= SIZE || y < 0 || y >= SIZE)
    return RES_REPEAT;

  int &cell = grid[y][x];

  if (cell == MISS || cell == HIT)
    return RES_REPEAT;

  if (cell == EMPTY) {
    cell = MISS;
    return RES_MISS;
  }

  if (cell == SHIP) {
    cell = HIT;
    shipsAlive--;
```

```
    if (shipsAlive <= 0)
      return RES_LOSE;
    return RES_HIT;
  }

  return RES_REPEAT;
}

void GameBoard::getBoardString(char *buffer, bool showShips) {
  int pos = 0;

  pos += sprintf(buffer + pos, "  0 1 2 3 4 5 6 7 8 9\n");
  pos += sprintf(buffer + pos, " ----------------------\n");

  for (int i = 0; i < SIZE; ++i) {
    pos += sprintf(buffer + pos, "%d ", i);
    for (int j = 0; j < SIZE; ++j) {
      char symbol = '.';
      int cell = grid[i][j];

      if (cell == EMPTY) {
        symbol = '.';
      } else if (cell == SHIP) {
        symbol = showShips ? '#' : '.';
      } else if (cell == MISS) {
        symbol = '*';
      } else if (cell == HIT) {
        symbol = 'X';
      }
      pos += sprintf(buffer + pos, "%c ", symbol);
    }
    pos += sprintf(buffer + pos, "\n");
  }
  buffer[pos] = '\0';
```

}

Остальной код (файлы ClientApp.cpp и ServerApp.cpp) находятся в репозитории https://github.com/BlackWLN/MAI_OS в папке cp1.

Протокол работы программы

Тестирование программы очень объемно из-за принципа работы игры «Морской бой» с выстрелами каждого из игроков(клиентов). Приводить его не стану для сокращения текста отчета.

Strace(через docker):

Strace-CLIENT:

vscode ➜ /workspaces/MAI_OS/cp1/build (main) $ strace ./client

execve("./client", ["./client"], 0xffffe2e4ad40 /* 35 vars */) = 0

brk(NULL) = 0xaaaaf9666000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff9fe69000

faccessat(AT_FDCWD, "/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=15048, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 15048, PROT_READ, MAP_PRIVATE, 3, 0) = 0xffff9fe65000

close(3) = 0

openat(AT_FDCWD, "/lib/aarch64-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\267\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2198944, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 2340896, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff9fa00000

mmap(0xffff9fa00000, 2275360, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffff9fa00000

munmap(0xffff9fc2c000, 63520) = 0

mprotect(0xffff9fc0b000, 65536, PROT_NONE) = 0

mmap(0xffff9fc1b000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x20b000) = 0xffff9fc1b000

mmap(0xffff9fc29000, 10272, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffff9fc29000

close(3) = 0

```
openat(AT_FDCWD, "/lib/aarch64-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\267\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=84296, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 213704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0xffff9fdff000
mmap(0xffff9fe00000, 148168, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffff9fe00000
munmap(0xffff9fdff000, 4096)          = 0
munmap(0xffff9fe25000, 58056)         = 0
mprotect(0xffff9fe14000, 61440, PROT_NONE) = 0
mmap(0xffff9fe23000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x13000) = 0xffff9fe23000
close(3)                              = 0
openat(AT_FDCWD, "/lib/aarch64-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\267\0\1\0\0\0\340u\2\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1637400, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 1805928, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0xffff9fc47000
mmap(0xffff9fc50000, 1740392, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffff9fc50000
munmap(0xffff9fc47000, 36864)         = 0
munmap(0xffff9fdf9000, 28264)         = 0
mprotect(0xffff9fdd8000, 61440, PROT_NONE) = 0
mmap(0xffff9fde7000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x187000) = 0xffff9fde7000
mmap(0xffff9fded000, 48744, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffff9fded000
close(3)                              = 0
openat(AT_FDCWD, "/lib/aarch64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=551064, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 680048, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0xffff9f959000
mmap(0xffff9f960000, 614512, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffff9f960000
munmap(0xffff9f959000, 28672)         = 0
```

```
munmap(0xffff9f9f7000, 32880)           = 0
mprotect(0xffff9f9e6000, 61440, PROT_NONE) = 0
mmap(0xffff9f9f5000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x85000) = 0xffff9f9f5000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0xffff9fe63000
set_tid_address(0xffff9fe63af0)         = 9388
set_robust_list(0xffff9fe63b00, 24)     = 0
rseq(0xffff9fe641c0, 0x20, 0, 0xd428bc00) = 0
mprotect(0xffff9fde7000, 16384, PROT_READ) = 0
mprotect(0xffff9f9f5000, 4096, PROT_READ) = 0
mprotect(0xffff9fe23000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0xffff9fe61000
mprotect(0xffff9fc1b000, 45056, PROT_READ) = 0
mprotect(0xaaaade044000, 4096, PROT_READ) = 0
mprotect(0xffff9fe6e000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
= 0
munmap(0xffff9fe65000, 15048)           = 0
getrandom("\x3f\xc9\x2b\x32\x76\x44\xf6\xa1", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0xaaaaf9666000
brk(0xaaaaf9687000)                     = 0xaaaaf9687000
futex(0xffff9fc297a4, FUTEX_WAKE_PRIVATE, 2147483647) = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0
write(1, "=== SEA FIGHT CLIENT ===\n", 25=== SEA FIGHT CLIENT ===
) = 25
write(1, "Enter your login: ", 18Enter your login: )      = 18
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0
read(0, pl1
"pl1\n", 1024)                  = 4
rt_sigaction(SIGRT_1, {sa_handler=0xffff9fcca6f0, sa_mask=[],
sa_flags=SA_ONSTACK|SA_RESTART|SA_SIGINFO}, NULL, 8) = 0
```

rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

mmap(NULL, 8454144, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0xffff9f000000

mprotect(0xffff9f010000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone(child_stack=0xffff9f80e940, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[9424], tls=0xffff9f80f8c0, child_tidptr=0xffff9f80f1d0) = 9424

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 0xffffc39d2f18) = 0

openat(AT_FDCWD, "/tmp/battleship_server_pipe", O_WRONLY) = 4

write(4, "\0\0\0\0pl1\0\0\0\0\0\0\0\0\0\210\237\346\237\377\377\0\0\0\0\0\0\377\377\0\0"..., 624) = 624

close(4)   = 0

clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},

[SERVER]: Welcome to the Sea Fight server!

>

Available games:

==========

No available games. Create your own with /create <game_name>

> 0xffffc39d2f18) = 0

write(1, "\nMain Menu\n", 11

Main Menu

)   = 11

write(1, "Commands:\n", 10Commands:

)   = 10

write(1, " /create <name>   - Create new "..., 37 /create <name>   - Create new game

) = 37

write(1, " /join <name>   - Join existi"..., 40 /join <name>   - Join existing game

) = 40

write(1, " /list   - Show availa"..., 42 /list   - Show available games

) = 42

write(1, " /stats   - Show your s"..., 42 /stats   - Show your statistics

```
) = 42
write(1, "  /quit           - Quit\n", 26  /quit           - Quit
) = 26
write(1, "> ", 2> )                     = 2
read(0, /list
"/list\n", 1024)              = 6
openat(AT_FDCWD, "/tmp/battleship_server_pipe", O_WRONLY) = 4
write(4, "\7\0\0\0pl1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 624) = 624
close(4)                              = 0
read(0,
Available games:
================
No available games. Create your own with /create <game_name>

> /stats
"/stats\n", 1024)             = 7
openat(AT_FDCWD, "/tmp/battleship_server_pipe", O_WRONLY) = 4
write(4, "\6\0\0\0pl1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 624) = 624
close(4)                              = 0
read(0,
Statistics for pl1:
================
Games played: 0
Wins: 0
Losses: 0
Win rate: 0%
Total shots: 0
Hits: 0
Accuracy: 0%

> /create game1
"/create game1\n", 1024)       = 14
openat(AT_FDCWD, "/tmp/battleship_server_pipe", O_WRONLY) = 4
write(4, "\1\0\0\0pl1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 624) = 624
```

close(4)   = 0

read(0,

[GAME]: Game 'game1' created! Waiting for opponent...

Use '/leave' to cancel

>

Available games:

==========

game1 (created by pl1)

To join game: /join <game_name>

> /leave

"/leave\n", 1024)   = 7

openat(AT_FDCWD, "/tmp/battleship_server_pipe", O_WRONLY) = 4

write(4, "\3\0\0\0pl1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 624) = 624

close(4)   = 0

write(1, "\nMain Menu\n", 11

Main Menu

)   = 11

write(1, "Commands:\n", 10Commands:

)   = 10

write(1, " /create <name>   - Create new "..., 37 /create <name>   - Create new game

) = 37

futex(0xffff9fded788, FUTEX_WAKE_PRIVATE, 1

[SERVER]: You left the game.

) = 1

> write(1, " /join <name>   - Join existi"..., 40 /join <name>   - Join existing game

) = 40

write(1, " /list   - Show availa"..., 42 /list   - Show available games

) = 42

futex(0xffff9fded788, FUTEX_WAKE_PRIVATE, 1

) = 1

Available games:

==========

No available games. Create your own with /create <game_name>

> write(1, " /stats   - Show your s"..., 42 /stats   - Show your statistics

) = 42

write(1, " /quit   - Quit\n", 26 /quit   - Quit

) = 26

write(1, "> ", 2>)   = 2

read(0, /quit

"/quit\n", 1024)   = 6

openat(AT_FDCWD, "/tmp/battleship_server_pipe", O_WRONLY) = 4

write(4, "\5\0\0\0pl1\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"..., 624) = 624

close(4)   = 0

rt_sigaction(SIGRTMIN, {sa_handler=0xffff9fccb780, sa_mask=[],
sa_flags=SA_RESTART|SA_SIGINFO}, NULL, 8) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

getpid()   = 9388

tgkill(9388, 9424, SIGRTMIN)   = 0

futex(0xffff9f80f874, FUTEX_WAKE_PRIVATE, 1) = 1

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

lseek(0, -1, SEEK_CUR)   = -1 ESPIPE (Illegal seek)

exit_group(0)   = ?

+++ exited with 0 +++

Strace-SERVER:

execve("./server", ["./server"], 0xffffff10feb30 /* 35 vars */) = 0

brk(NULL)   = 0xaaaad800d000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0xffffa0575000

faccessat(AT_FDCWD, "/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=15048, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 15048, PROT_READ, MAP_PRIVATE, 3, 0) = 0xffffa0571000

close(3)   = 0

openat(AT_FDCWD, "/lib/aarch64-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\267\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2198944, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 2340896, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0xffffa0200000
mmap(0xffffa0200000, 2275360, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffffa0200000
munmap(0xffffa042c000, 63520)           = 0
mprotect(0xffffa040b000, 65536, PROT_NONE) = 0
mmap(0xffffa041b000, 57344, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x20b000) = 0xffffa041b000
mmap(0xffffa0429000, 10272, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffffa0429000
close(3)                                = 0
openat(AT_FDCWD, "/lib/aarch64-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\267\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=84296, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 213704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0xffffa050b000
mmap(0xffffa0510000, 148168, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffffa0510000
munmap(0xffffa050b000, 20480)           = 0
munmap(0xffffa0535000, 41672)           = 0
mprotect(0xffffa0524000, 61440, PROT_NONE) = 0
mmap(0xffffa0533000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x13000) = 0xffffa0533000
close(3)                                = 0
openat(AT_FDCWD, "/lib/aarch64-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\267\0\1\0\0\0\340u\2\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1637400, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 1805928, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0xffffa0047000
mmap(0xffffa0050000, 1740392, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffffa0050000
munmap(0xffffa0047000, 36864)           = 0
munmap(0xffffa01f9000, 28264)           = 0
mprotect(0xffffa01d8000, 61440, PROT_NONE) = 0
```

mmap(0xffffa01e7000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x187000) = 0xffffa01e7000

mmap(0xffffa01ed000, 48744, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffffa01ed000

close(3) = 0

openat(AT_FDCWD, "/lib/aarch64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=551064, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 680048, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffffa0469000

mmap(0xffffa0470000, 614512, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffffa0470000

munmap(0xffffa0469000, 28672) = 0

munmap(0xffffa0507000, 32880) = 0

mprotect(0xffffa04f6000, 61440, PROT_NONE) = 0

mmap(0xffffa0505000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x85000) = 0xffffa0505000

close(3) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffffa056f000

set_tid_address(0xffffa056faf0) = 9288

set_robust_list(0xffffa056fb00, 24) = 0

rseq(0xffffa05701c0, 0x20, 0, 0xd428bc00) = 0

mprotect(0xffffa01e7000, 16384, PROT_READ) = 0

mprotect(0xffffa0505000, 4096, PROT_READ) = 0

mprotect(0xffffa0533000, 4096, PROT_READ) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffffa056d000

mprotect(0xffffa041b000, 45056, PROT_READ) = 0

mprotect(0xaaaac1630000, 4096, PROT_READ) = 0

mprotect(0xffffa057a000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0xffffa0571000, 15048) = 0

getrandom("\x09\x69\x05\xf3\x5f\x40\xa4\xc4", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0xaaaad800d000

```
brk(0xaaaad802e000)                     = 0xaaaad802e000
futex(0xffffa04297a4, FUTEX_WAKE_PRIVATE, 2147483647) = 0
unlinkat(AT_FDCWD, "/tmp/battleship_server_pipe", 0) = 0
mknodat(AT_FDCWD, "/tmp/battleship_server_pipe", S_IFIFO|0666) = 0
openat(AT_FDCWD, "/tmp/battleship_server_pipe", O_RDWR) = 3
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...},
AT_EMPTY_PATH) = 0
write(1, "Server running. Waiting...\n", 27Server running. Waiting...
) = 27
read(3, "\0\0\0\0pl1\0\0\0\0\0\0\0\0\0\210\237\346\237\377\377\0\0\0\0\0\0\377\377\0\0"..., 624)
= 624
write(1, "[Login] New player: pl1\n", 24[Login] New player: pl1
) = 24
openat(AT_FDCWD, "/tmp/client_pl1", O_WRONLY) = 4
write(4, "\10\0\0\0\377\377\0\0\0\0\0\0\0\0\0\0l1\0\377\377\0\0\340\1W\240\377\377\0\0"...,
624) = 624
close(4)                                = 0
openat(AT_FDCWD, "/tmp/client_pl1", O_WRONLY) = 4
write(4, "\t\0\0\0SERVER\0\240\377\377\0\0#\320\6\240\377\377\0\0\0\0\0\0\0\0\0"..., 624) =
624
close(4)                                = 0
read(3, "\7\0\0\0pl1\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"..., 624) = 624
openat(AT_FDCWD, "/tmp/client_pl1", O_WRONLY) = 4
write(4, "\t\0\0\0SERVER\0\0\0\0\0\0\0l1\0\377\377\0\0\340\1W\240\377\377\0\0"..., 624) = 624
close(4)                                = 0
read(3, "\6\0\0\0pl1\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"..., 624) = 624
openat(AT_FDCWD, "/tmp/client_pl1", O_WRONLY) = 4
write(4, "\17\0\0\0SERVER\0\0\0\0\0\0pl1\0\377\377\0\0\340\1W\240\377\377\0\0"..., 624) =
624
close(4)                                = 0
read(3, "\1\0\0\0pl1\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"..., 624) = 624
write(1, "[Game Created] game1 by pl1\n", 28[Game Created] game1 by pl1
) = 28
openat(AT_FDCWD, "/tmp/client_pl1", O_WRONLY) = 4
```

write(4, "\n\0\0\0\377\377\0\0\3\0\0\0\0\0\0\0pl1\0\377\377\0\0\340\1W\240\377\377\0\0"..., 624) = 624

close(4) = 0

openat(AT_FDCWD, "/tmp/client_pl1", O_WRONLY) = 4

write(4, "\t\0\0\0SERVER\0\365\377\377\0\0\0\0\0\0\0\0\0\0@s\335\365\377\377\0\0"..., 624) = 624

close(4) = 0

read(3, "\3\0\0\0pl1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 624) = 624

write(1, "[Game Cancelled] pl1 cancelled g"..., 37[Game Cancelled] pl1 cancelled game1

) = 37

openat(AT_FDCWD, "/tmp/client_pl1", O_WRONLY) = 4

write(4, "\10\0\0\0\377\377\0\0\3\0\0\0\0\0\0\0pl1\0\377\377\0\0\340\1W\240\377\377\0\0"..., 624) = 624

close(4) = 0

openat(AT_FDCWD, "/tmp/client_pl1", O_WRONLY) = 4

write(4, "\t\0\0\0SERVER\0 /join <game_name>\n\0\0"..., 624) = 624

close(4) = 0

read(3, "\5\0\0\0pl1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 624) = 624

write(1, "[Logout] Player pl1 removed from"..., 48[Logout] Player pl1 removed from server's list.

) = 48

read(3, 0xfffff5dd7818, 624) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---

read(3, 0xfffff5dd7818, 624) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---

read(3, 0xfffff5dd7818, 624) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---

read(3, 0xfffff5dd7818, 624) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---

read(3, 0xfffff5dd7818, 624) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---

read(3, 0xfffff5dd7818, 624)	= ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---

read(3, 0xfffff5dd7818, 624)	= ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---

read(3, 0xfffff5dd7818, 624)	= ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---

read(3, 0xfffff5dd7818, 624)	= ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---

read(3, 0xfffff5dd7818, 624)	= ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---

read(3, ^Z

[2]+ Stopped	strace ./server

Вывод

В ходе выполнения лабораторной работы были приобретены и закреплены практические навыки системного программирования в операционной системе Linux, а именно проектирование и реализация межпроцессного взаимодействия (IPC).

Были решены следующие задачи:

• Освоение Named Pipes: Изучен механизм работы с именованными каналами (mkfifo, open, read, write). Реализована надежная схема обмена данными, предотвращающая блокировки (deadlocks) за счет правильного выбора режимов открытия файлов (O_RDWR на сервере) и архитектуры «один общий канал на вход — уникальные каналы на выход».

• Сетевая архитектура: Реализована топология «Звезда», где сервер выступает в роли координатора и арбитра. Это позволило централизовать логику игры, упростить клиенты до уровня терминалов ввода-вывода и исключить возможность жульничества (так как карта противника хранится только на сервере).

• Многопоточное программирование: Применение POSIX Threads позволило реализовать асинхронный ввод-вывод на клиенте, разделив задачи ожидания сети и взаимодействия с пользователем, что является стандартом для современных сетевых приложений.

• Декомпозиция и модульность: Проект разделен на логические модули (сетевая обертка, протокол, игровая логика), что упрощает его поддержку и масштабирование.

• Разработанный программный комплекс полностью удовлетворяет требованиям задания, демонстрирует стабильную работу при сценарии игры двух пользователей и подтверждает

эффективность выбранных методов IPC для решения задач локального клиент-серверного взаимодействия