

ВВЕДЕНИЕ

Целью данного проекта является разработка программного средства для игры "Крестики-нолики" с использованием клиент-серверной архитектуры. Игра "Крестики-нолики" является популярной логической игрой, где два игрока поочередно ставят свои символы (крестик и нолик) на игровом поле размером 3x3. Победителем становится игрок, который первым соберет свои символы в ряд по горизонтали, вертикали или диагонали. Для реализации игры была выбрана клиент-серверная архитектура, которая позволяет нескольким игрокам подключаться к серверу и играть между собой. Сервер управляет игровым процессом, а клиенты отправляют свои ходы на сервер и получают состояние игры.

В данной пояснительной записке представлено описание разработанного программного средства. Анализируется предметная область игры "Крестики-нолики" и описывается структура функционирования программы. Также приводится описание созданных программных конструкций, основных функций и проверка работы программы.

Целью данного проекта является разработка удобного и эффективного программного средства для игры "Крестики-нолики", которое может быть использовано для развлечения и развития логического мышления игроков..

Целью курсовой работы является создание программы, реализующей такой функционал в системе Linux.

Для полного выполнения курсовой работы необходимо выполнить ряд задач:

- описать предметную область;
- проанализировать изучаемые процессы;
- выполнить программную реализацию;
- выполнить тестирование проектных решений:

2 СТРУКТУРА ФУНКЦИОНИРОВАНИЯ ПРОГРАММЫ

Программа разделена на несколько файлов, каждый из которых выполняет определенные функции. Основные файлы программы включают:

- **main.c**: В этом файле содержится главная функция **main**, которая является точкой входа в программу. В функции **main** определяется режим работы программы (сервер или клиент) и вызываются соответствующие функции для запуска сервера или клиента;

- **server.c** и **client.c**: Эти файлы содержат реализацию игры на стороне сервера и клиента соответственно. Функции в этих файлах отвечают за установку соединения, передачу данных и управление игровым процессом. Также эти файлы содержат функции, отвечающие за работу с сетевыми соединениями. Здесь реализованы функции для создания сокетов, установки соединения и передачи данных между клиентом и сервером

- **game_logic.c** и **game_logic.h**: Эти файлы содержат реализацию игровой логики. Здесь определены функции, отвечающие за проверку правил игры, обновление состояния игры и определение победителя.

При запуске программы в функции **main** определяется режим работы: сервер или клиент.

В режиме сервера, программа создает сокет сервера, ожидает подключения клиента и управляет ходом игры, взаимодействуя с клиентом через сокет.

В режиме клиента, программа создает сокет клиента, подключается к серверу и принимает команды от сервера для игры.

Реализация игры на стороне сервера и клиента основана на взаимодействии между сервером и клиентом через сетевое соединение.

Функции в файле **game_logic.c** отвечают за проверку правил игры, обновление состояния игры и определение победителя.

Такая структура позволяет эффективно управлять игровым процессом, обеспечивая взаимодействие между сервером и клиентом и обработку игровой логики.

3 ОПИСАНИЕ СОЗДАННЫХ ПРОГРАММНЫХ КОНСТРУКЦИЙ

3.1 Структура игры

Игра "Крестики-нолики" представляет собой классическую игру для двух игроков. Она состоит из следующих элементов:

- Игровое поле размером 3x3, представленное двумерным массивом типа **char**. Каждая ячейка может содержать символы 'X', 'O' или пустое значение ''.
- Переменные **row** и **col**, которые хранят индексы строки и столбца для выполнения хода игрока.
- Переменная **player**, определяющая текущего игрока. Значение 1 соответствует первому игроку, а значение 2 - второму игроку.
- Переменная **moves**, содержащая количество сделанных ходов.
- Переменная **total_moves**, хранящая общее количество доступных ходов.
- Массив **message** типа **char**, используемый для передачи сообщений между клиентом и сервером.
- Переменная **bytes_read**, которая хранит количество прочитанных байтов из сокета.
- Переменная **board_size**, определяющая размер игрового поля.
- Переменная **mode**, которая указывает режим работы программы: клиент или сервер.
- Переменные **server_fd** и **new_socket**, представляющие сокеты для сервера и нового соединения соответственно.
- Структура **address**, содержащая информацию о сокете.
- Переменная **addrlen**, хранящая размер структуры **address**.
- Перечисление **GameResult**, определяющее возможные результаты игры.

3.2 Инициализация игры

Инициализация игры является важным шагом, который подготавливает игровое поле и переменные для дальнейшего хода игры. В данном разделе можно провести следующие действия:

- Создание сокетов для клиента и сервера с использованием функции **socket()**. Сокеты представляют собой точки соединения для обмена данными между клиентом и сервером.
- Установление соединения между клиентом и сервером. Для этого клиент должен вызвать функцию **connect()**, передавая адрес сервера, в то время как сервер должен вызвать функции **bind()** и **listen()**, чтобы прослушивать входящие соединения.
- Инициализация игрового поля **board** пустыми значениями. Это достигается путем заполнения каждой ячейки массива символом пробела (' ').

- Установка начального значения переменной **player**. В данной реализации игры, игрок 1 всегда начинает первым. Таким образом, переменная **player** устанавливается равной 1.
- Установка начальных значений переменных **moves** и **total_moves**. Переменная **moves** отслеживает количество сделанных ходов, которые будут использоваться для определения окончания игры. Переменная **total_moves** определяет общее количество доступных ходов на игровом поле.
- Передача сообщения о начале игры между клиентом и сервером. Для этого используются функции **write()** и **read()**. Клиент отправляет серверу сообщение о готовности к игре, а сервер ожидает это сообщение и подтверждает готовность к началу игры.

4 ОПИСАНИЕ ОСНОВНЫХ ФУНКЦИЙ И ПЕРЕМЕННЫХ

4.1 Описание функций и переменных файла `main.c`

– **int main()** - это основная функция программы, которая выполняет управление ходом игры. Она содержит цикл, в котором происходит взаимодействие с игроками, проверка состояния игры, вызов других функций и вывод информации на экран.

– **int mode** - это переменная типа `int`, которая указывает на режим игры или другие параметры, связанные с логикой игры. Конкретное значение и его назначение будут определяться в коде программы.

– **int server_fd, new_socket** - это переменные типа `int`, которые используются для работы с сокетами в клиент-серверной модели.

– **server_fd** обычно представляет файловый дескриптор серверного сокета, а **new_socket** - файловый дескриптор нового сокета, устанавливаемого при установке соединения с клиентом.

– **struct sockaddr_in address** - это структура, используемая для хранения информации о сетевом адресе. В данном случае, она представляет адрес сервера.

– **int addrlen = sizeof(address)** - это переменная типа `int`, которая хранит размер структуры **address**. Она используется при вызове некоторых функций для передачи размера структуры.

4.2 Описание функций и переменных файла server.c

– **void play_game_server(int client_socket)** - это функция, которая отвечает за игровой процесс для сервера. Она принимает аргумент **client_socket**, который представляет сокет для связи с клиентом. Внутри функции происходит обмен данными между сервером и клиентом, передача ходов и обновление состояния игры.

– **char board[3][3]** - это двумерный символьный массив, который представляет игровое поле крестики-нолики. Он состоит из 3 строк и 3 столбцов и используется для отображения текущего состояния игры.

– **int row, col** - это переменные типа **int**, используемые для хранения текущих координат (строки и столбца) на игровом поле. Они используются для указания местоположения игрока при ходе.

– **int player = 1** - это переменная типа **int**, которая указывает на текущего игрока. В данном случае, значение 1 представляет первого игрока, а другое значение, например, 2, может представлять второго игрока.

– **int moves = 0** - это переменная типа **int**, которая отслеживает количество сделанных ходов в игре. Она увеличивается каждый раз, когда игрок совершает ход.

– **int total_moves = 0** - это переменная типа **int**, которая отслеживает общее количество сделанных ходов в игре. В отличие от **moves**, **total_moves** не обнуляется после каждого хода и продолжает увеличиваться до конца игры.

– **char message[256]** - это символьный массив размером 256, используемый для хранения сообщений или данных, полученных из сети или других источников.

4.3 Описание функций и переменных файла `client.c`

– **`void play_game_client(int server_socket)`** - это функция, которая отвечает за игровой процесс для клиента. Она принимает аргумент `server_socket`, который представляет сокет для связи с сервером. Внутри функции происходит обмен данными между клиентом и сервером, передача ходов и обновление состояния игры.

– **`char board[3][3]`** - это двумерный символьный массив, который представляет игровое поле крестики-нолики. Он состоит из 3 строк и 3 столбцов и используется для отображения текущего состояния игры.

`int row, col` - это переменные типа `int`, используемые для хранения текущих координат (строки и столбца) на игровом поле. Они используются для указания местоположения игрока при ходе.

`int player = 1` - это переменная типа `int`, которая указывает на текущего игрока. В данном случае, значение 1 представляет первого игрока, а другое значение, например, 2, может представлять второго игрока.

– **`int moves = 0`** - это переменная типа `int`, которая отслеживает количество сделанных ходов в игре. Она увеличивается каждый раз, когда игрок совершает ход.

– **`int total_moves = 0`** - это переменная типа `int`, которая отслеживает общее количество сделанных ходов в игре. В отличие от `moves`, `total_moves` не обнуляется после каждого хода и продолжает увеличиваться до конца игры.

– **`char message[256]`** - это символьный массив размером 256, используемый для хранения сообщений или данных, полученных из сети или других источников.

– **`int bytes_read = read(server_socket, message, sizeof(message))`** - это переменная типа `int`, которая хранит количество байтов, считанных из сокета `server_socket` и сохраненных в массиве `message`. Функция `read` используется для чтения данных из сокета.

4.4 Описание функций и переменных в файле `game_logic.c`

– **`int check_winner(char board[3][3])`** - это функция, которая проверяет, есть ли победитель в текущем состоянии игрового поля. Она принимает аргумент `board`, который представляет игровое поле размером 3x3. Функция проверяет все возможные комбинации выигрышных позиций на игровом поле и возвращает результат проверки: 0 - нет победителя, 1 - победа первого игрока, 2 - победа второго игрока.

– **int make_move(char board[3][3], int row, int col, int player)** - это функция, которая выполняет ход игрока на указанную позицию на игровом поле. Она принимает аргументы board (игровое поле), row (строка), col (столбец) и player (игрок). Функция обновляет состояние игрового поля, записывая символ игрока (крестик или нолик) в соответствующую ячейку, и возвращает результат хода: 0 - ход совершен успешно, 1 - позиция уже занята другим игроком, 2 - недопустимая позиция.

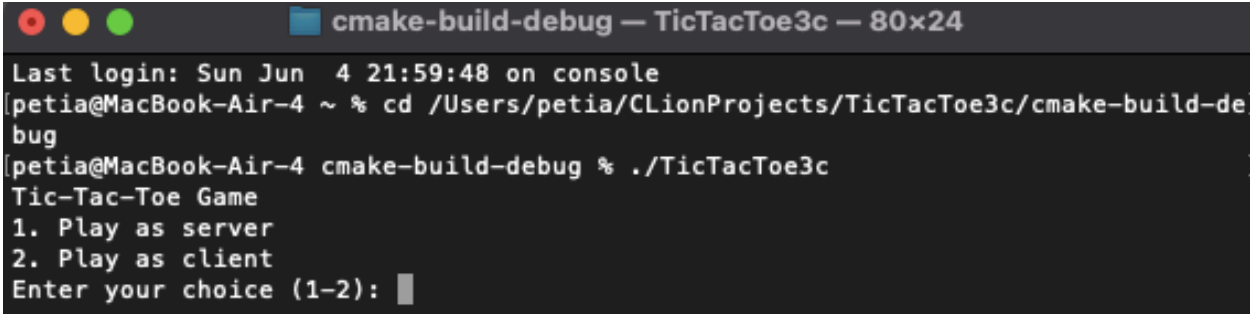
5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

5.1 Правила игры

1. Игра начинается с пустого игрового поля размером 3x3.
2. Игроки ходят поочередно, ставя свои символы на свободные клетки поля.
3. Первый игрок использует символ "X", а второй игрок - символ "O".
4. Чтобы сделать ход, введите номер строки и номер столбца, где вы хотите поставить свой символ. Нумерация начинается с 0.
5. Игра продолжается до тех пор, пока не будет достигнута одна из следующих ситуаций:
6. Один из игроков собрал свои символы в ряд по вертикали, горизонтали или диагонали. В этом случае игрок, сделавший победный ход, объявляется победителем.
7. Все клетки поля заполнены, и ни один из игроков не собрал свои символы в ряд. В этом случае игра считается ничьей.
8. По окончании игры вы можете начать новую игру или выйти из программы.

5.2 Запуск игры

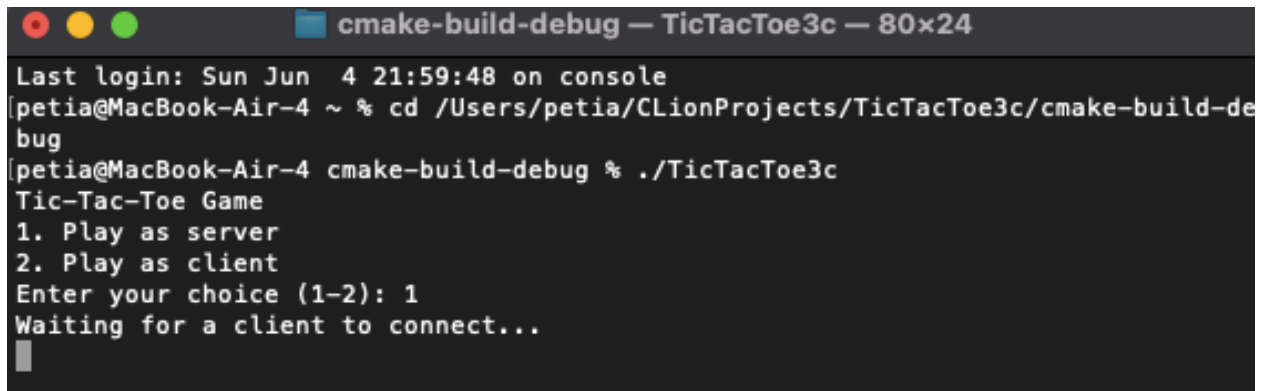
После компиляции и сборки программного проекта с помощью `make` пользователь должен открыть терминал в папке с исполняемым файлом с названием "TicTacToe". Для запуска программы пользователь должен ввести в терминале `./TicTacToe`.



```
cmake-build-debug — TicTacToe3c — 80x24
Last login: Sun Jun  4 21:59:48 on console
[petia@MacBook-Air-4 ~ % cd /Users/petia/CLionProjects/TicTacToe3c/cmake-build-de
bug
[petia@MacBook-Air-4 cmake-build-debug % ./TicTacToe3c
Tic-Tac-Toe Game
1. Play as server
2. Play as client
Enter your choice (1-2): █
```

Рисунок 5.1 - пример команды для запуска игры

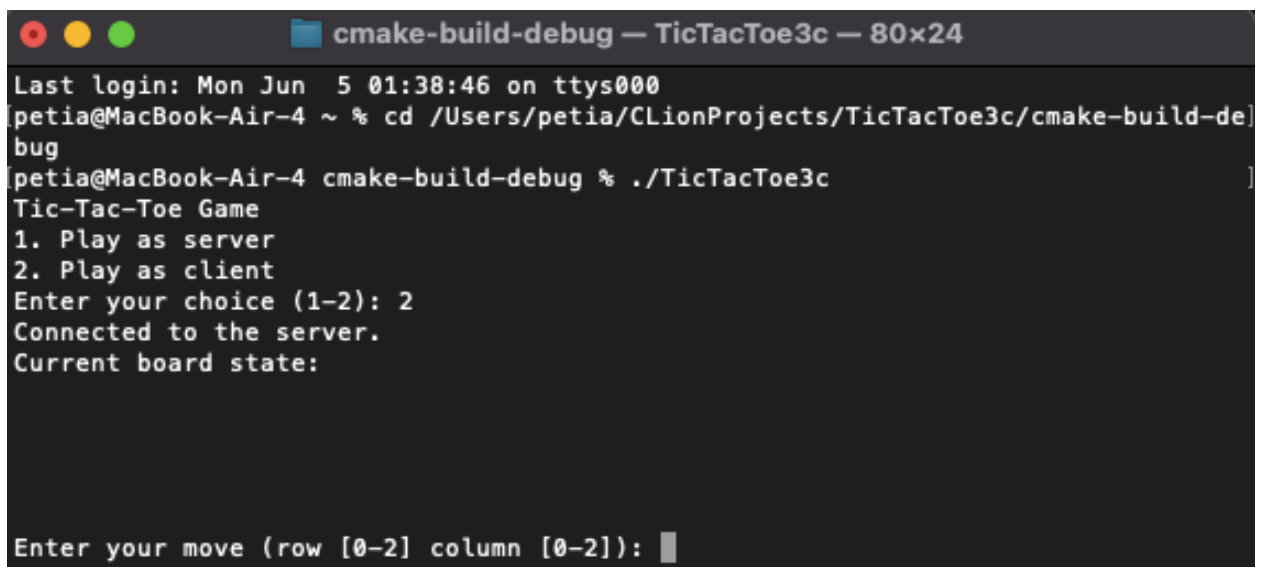
После запуска игры пользователь увидит меню с выбором на чьей стороне играть, на стороне сервера или на стороне клиента. Для корректной работы программы необходимо в первом окне терминала выбрать игру на стороне сервера.



```
cmake-build-debug — TicTacToe3c — 80x24
Last login: Sun Jun  4 21:59:48 on console
[petia@MacBook-Air-4 ~ % cd /Users/petia/CLionProjects/TicTacToe3c/cmake-build-de
bug
[petia@MacBook-Air-4 cmake-build-debug % ./TicTacToe3c
Tic-Tac-Toe Game
1. Play as server
2. Play as client
Enter your choice (1-2): 1
Waiting for a client to connect...
█
```

Рисунок 5.2 - меню с выбором стороны игрока

Теперь пользователь видит окно с ожиданием подключения клиента. Далее следует во втором окне терминала выбрать игру на стороне клиента.



```
cmake-build-debug — TicTacToe3c — 80x24
Last login: Mon Jun  5 01:38:46 on ttys000
[petia@MacBook-Air-4 ~ % cd /Users/petia/CLionProjects/TicTacToe3c/cmake-build-de
bug
[petia@MacBook-Air-4 cmake-build-debug % ./TicTacToe3c
Tic-Tac-Toe Game
1. Play as server
2. Play as client
Enter your choice (1-2): 2
Connected to the server.
Current board state:

Enter your move (row [0-2] column [0-2]): █
```

Рисунок 5. - игра на стороне клиента

Теперь во втором окне терминала отображается текущее состояние поля на стороне клиента и игроку предлагается сделать ход. Далее можно переходить непосредственно к самой игре.

ЗАКЛЮЧЕНИЕ

В данной курсовой работе была разработана сетевая игра "крестики-нолики" в операционной системе Linux. Целью проекта было создание функционального и надежного программного средства, которое позволяло бы играть в игру "крестики-нолики" между несколькими игроками через сеть.

В процессе разработки программного средства была реализована клиент-серверная модель взаимодействия, которая обеспечивает совместную игру нескольких игроков. Это требовало проектирования и реализации сетевого взаимодействия между сервером и клиентами, обмена данными и синхронизации игрового процесса. Были изучены принципы работы с сокетами, протоколами передачи данных и обработки сетевых соединений.

В ходе работы была проведена обширная литературная ревизия по теме проекта и изучены особенности создания программ под систему Linux. Были изучены принципы многопоточности, синхронизации и обработки сетевых событий в контексте игрового приложения. Это позволило разработать эффективные и безопасные алгоритмы для обработки ходов игроков, проверки правильности действий и обновления состояния игры.

Для обеспечения качества программного средства было проведено тестирование программы. Были созданы различные тестовые сценарии, которые покрывали различные аспекты игрового процесса и возможные ситуации. Тестирование помогло выявить и исправить ошибки, а также проверить работоспособность программы в различных условиях.

В процессе разработки также были проведены измерения временных затрат и затрат памяти для выполнения операций копирования и скачивания данных. Это помогло оптимизировать код программы и улучшить ее производительность.

В результате выполнения данной курсовой работы была успешно разработана сетевая игра "крестики-нолики" в операционной системе Linux. Разработанное программное средство обладает функциональностью, надежностью и эффективностью, позволяя играть в игру "крестики-нолики" между несколькими игроками через сеть.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Командная строка Linux. Полное руководство./ Уильям Шоттс – Санкт-Петербург, 2022. – 544 с
- [2] Linux. Системное программирование./ Роберт Лав – Санкт-Петербург, 2014. – 554 с
- [3] Нейросеть ChatGPT [Электронный ресурс]. – Режим доступа: <https://chat.openai.com> – Дата доступа 05.05.2023.
- [4] Программирование в Linux [Электронный ресурс]. – Режим доступа: <https://www.linuxlib.ru/prog/> – Дата доступа: 04.05.2023.
- [5] Внутреннее устройство Linux / Брайан Уорд – Санкт-Петербург, 2022. – 480 с
- [6] Программирование в сетях UNIX. Том 1: Сокеты./ У. Ричард Стивенс, Билл Феннер, Эндрю М. Рудофф – Санкт-Петербург, 2015. – 1024 с
- [7] Linux. Программирование интерфейсов системы./ М. Керриск – Санкт-Петербург, 2011. – 1520 с
- [8] Linux. Серверные приложения и веб-сервер Apache./ А. Молоховец – Санкт-Петербург, 2013. – 368 с
- [9] Программирование на языке C для Linux./ Х. Блейздель – Санкт-Петербург, 2014. - 768 с
- [10] Beej's Guide to Network Programming. [Электронный ресурс]. – Режим доступа: <https://beej.us/guide/bgnet> – Дата доступа: 06.05.2023.