

Курсовая работа: Программная реализация сетевого сервера.  
Чернышев Ринат 24ПТ2  
1.0

Создано системой Doxygen 1.9.1



1 Алфавитный указатель классов	1
1.1 Классы	1
2 Список файлов	3
2.1 Файлы	3
3 Классы	5
3.1 Класс Server	5
3.1.1 Подробное описание	6
3.1.2 Конструктор(ы)	6
3.1.2.1 Server()	6
3.1.3 Методы	6
3.1.3.1 authenticate()	6
3.1.3.2 calculateSumOfSquares()	7
3.1.3.3 generateSalt()	8
3.1.3.4 handleClient()	8
3.1.3.5 loadUserDatabase()	8
3.1.3.6 logError()	9
3.1.3.7 processVectors()	9
3.1.3.8 readExact()	10
3.1.3.9 sha224Hash()	10
3.1.3.10 start()	11
3.1.4 Данные класса	11
3.1.4.1 logPath	11
3.1.4.2 port	11
3.1.4.3 userDbPath	12
3.1.4.4 users	12
4 Файлы	13
4.1 Файл main.cpp	13
4.1.1 Подробное описание	14
4.1.2 Функции	14
4.1.2.1 main()	14
4.1.2.2 showHelp()	14
4.2 Файл server.cpp	15
4.2.1 Подробное описание	15
4.3 Файл server.h	15
4.3.1 Подробное описание	16
Предметный указатель	17



# Глава 1

## Алфавитный указатель классов

### 1.1 Классы

Классы с их кратким описанием.

<a href="#">Server</a>	Класс сервера для обработки клиентских подключений . . . . .	<a href="#">5</a>
------------------------	--	-------------------



## Глава 2

# Список файлов

### 2.1 Файлы

Полный список файлов.

<a href="#">main.cpp</a>	Точка входа серверного приложения . . . . .	13
<a href="#">server.cpp</a>	Реализация методов класса <a href="#">Server</a> для обработки клиентских подключений . . . .	15
<a href="#">server.h</a>	Заголовочный файл класса <a href="#">Server</a> . . . . .	15





## Глава 3

# Классы

### 3.1 Класс Server

Класс сервера для обработки клиентских подключений.

```
#include <server.h>
```

#### Открытые члены

- `Server` (int `port`, const std::string &`userDbPath`, const std::string &`logPath`)  
Конструктор сервера.
- bool `start` ()  
Запускает сервер и начинает прослушивание порта.

#### Закрытые члены

- void `logError` (const std::string &`message`, bool `isCritical`)  
Записывает сообщение об ошибке в журнал.
- void `loadUserDatabase` ()  
Загружает базу данных пользователей из файла.
- std::string `sha224Hash` (const std::string &`input`)  
Вычисляет SHA-224 хэш строки.
- std::string `generateSalt` ()  
Генерирует случайную соль для аутентификации.
- void `handleClient` (int `clientSocket`)  
Обрабатывает подключение клиента.
- bool `authenticate` (int `clientSocket`)  
Аутентифицирует клиента по протоколу SHA-224.
- void `processVectors` (int `clientSocket`)  
Обрабатывает передачу векторов от аутентифицированного клиента.
- int16\_t `calculateSumOfSquares` (const std::vector< int16\_t > &`vector`)  
Вычисляет сумму квадратов элементов вектора.
- bool `readExact` (int `socket`, void \*`buffer`, size\_t `size`)  
Читает точное количество байт из сокета.

## Закрытые данные

- `int port`  
Порт сервера
- `std::string userDbPath`  
Путь к базе пользователей
- `std::string logPath`  
Путь к файлу журнала
- `std::unordered_map< std::string, std::string > users`  
Кэш пользователей (логин:пароль)

### 3.1.1 Подробное описание

Класс сервера для обработки клиентских подключений.

Обеспечивает сетевую коммуникацию, аутентификацию пользователей по протоколу SHA-224 с генерацией соли на сервере, прием и обработку векторных данных в двоичном формате, а также логирование.

### 3.1.2 Конструктор(ы)

#### 3.1.2.1 Server()

```
Server::Server (
    int port,
    const std::string & userDbPath,
    const std::string & logPath )
```

Конструктор сервера.

Конструктор класса `Server`.

Аргументы

<code>port</code>	Порт для прослушивания подключений.
<code>userDbPath</code>	Путь к файлу базы данных пользователей.
<code>logPath</code>	Путь к файлу журнала сервера.

### 3.1.3 Методы

#### 3.1.3.1 authenticate()

```
bool Server::authenticate (
    int clientSocket ) [private]
```

Аутентифицирует клиента по протоколу SHA-224.

Аутентифицирует клиента по протоколу SHA-224 с солью.

Аргументы

clientSocket	Дескриптор сокета клиента.
--------------	----------------------------

Возвращает

true если аутентификация успешна.

Протокол:

1. Клиент отправляет логин
2. Сервер генерирует и отправляет соль (16 hex символов)
3. Клиент отправляет HASH(SALT || PASSWORD)
4. Сервер проверяет хэш

Аргументы

clientSocket	Дескриптор сокета клиента.
--------------	----------------------------

Возвращает

true если аутентификация успешна, false в противном случае.

### 3.1.3.2 calculateSumOfSquares()

```
int16_t Server::calculateSumOfSquares (
    const std::vector< int16_t > & vector ) [private]
```

Вычисляет сумму квадратов элементов вектора.

Вычисляет сумму квадратов элементов вектора с проверкой переполнения.

Аргументы

vector	Вектор 16-битных целых чисел для обработки.
--------	---

Возвращает

Сумма квадратов элементов.

При переполнении вверх возвращает 32767 ( $2^{15}$ ), при переполнении вниз возвращает -32768 ( $-2^{15}$ ).

Аргументы

vector	Вектор 16-битных целых чисел.
--------	-------------------------------

Возвращает

Сумма квадратов элементов вектора.

При переполнении вверх возвращает 32767 ( $2^{15}$ ), при переполнении вниз возвращает -32768 ( $-2^{15}$ ).

### 3.1.3.3 generateSalt()

```
std::string Server::generateSalt ( ) [private]
```

Генерирует случайную соль для аутентификации.

Возвращает

Соль в виде строки из 16 шестнадцатеричных символов (64 бита).

### 3.1.3.4 handleClient()

```
void Server::handleClient (
    int clientSocket ) [private]
```

Обрабатывает подключение клиента.

Обрабатывает подключение одного клиента.

Аргументы

clientSocket	Дескриптор сокета клиента для обмена данными.
clientSocket	Дескриптор сокета клиента.

### 3.1.3.5 loadUserDatabase()

```
void Server::loadUserDatabase ( ) [private]
```

Загружает базу данных пользователей из файла.

Формат файла: каждая строка содержит "логин:пароль".

Читает файл построчно, парсит строки формата "логин:пароль" и сохраняет данные во внутреннем контейнере users.

#### 3.1.3.6 logError()

```
void Server::logError (
    const std::string & message,
    bool isCritical ) [private]
```

Записывает сообщение об ошибке в журнал.

Записывает сообщение об ошибке в файл журнала.

Аргументы

message	Текст сообщения об ошибке.
isCritical	Флаг критичности ошибки (true для критических).
message	Текст сообщения об ошибке.
isCritical	Флаг критичности ошибки.

#### 3.1.3.7 processVectors()

```
void Server::processVectors (
    int clientSocket ) [private]
```

Обрабатывает передачу векторов от аутентифицированного клиента.

Аргументы

clientSocket	Дескриптор сокета клиента для обмена данными.
--------------	---

Ожидает данные в двоичном формате:

- количество векторов (uint32\_t)
- для каждого вектора:
  - размер (uint32\_t)
  - данные (int16\_t[]) Отправляет результаты в двоичном формате:
- количество результатов (uint32\_t)
- результаты (int16\_t[])

Аргументы

clientSocket	Дескриптор сокета клиента.
--------------	----------------------------

Ожидает данные в двоичном формате согласно ТЗ.

### 3.1.3.8 readExact()

```
bool Server::readExact (
    int socket,
    void * buffer,
    size_t size ) [private]
```

Читает точное количество байт из сокета.

Вспомогательная функция для точного чтения из сокета.

Аргументы

socket	Дескриптор сокета для чтения.
buffer	Буфер для принятых данных.
size	Количество байт для чтения.

Возвращает

true если все байты прочитаны, false при ошибке.

Аргументы

socket	Дескриптор сокета.
buffer	Буфер для данных.
size	Количество байт для чтения.

Возвращает

true если все байты прочитаны, false при ошибке.

### 3.1.3.9 sha224Hash()

```
std::string Server::sha224Hash (
    const std::string & input ) [private]
```

Вычисляет SHA-224 хэш строки.

Вычисляет SHA-224 хэш для входной строки.

Аргументы

input	Входная строка для хэширования.
-------	---------------------------------

Возвращает

SHA-224 хэш в шестнадцатеричном формате (56 символов, верхний регистр).

Аргументы

input	Входная строка для хэширования.
-------	---------------------------------

Возвращает

SHA-224 хэш строки в шестнадцатеричном формате (верхний регистр, 56 символов).

#### 3.1.3.10 start()

bool Server::start ( )

Запускает сервер и начинает прослушивание порта.

Запускает основной цикл работы сервера.

Возвращает

true если сервер успешно запущен, false при ошибке.

true если сервер успешно запущен, false при критической ошибке.

#### 3.1.4 Данные класса

##### 3.1.4.1 logPath

std::string Server::logPath [private]

Путь к файлу журнала

##### 3.1.4.2 port

int Server::port [private]

Порт сервера

#### 3.1.4.3 userDbPath

`std::string Server::userDbPath` [private]

Путь к базе пользователей

#### 3.1.4.4 users

`std::unordered_map<std::string, std::string> Server::users` [private]

Кэш пользователей (логин:пароль)

Объявления и описания членов классов находятся в файлах:

- [server.h](#)
- [server.cpp](#)



## Глава 4

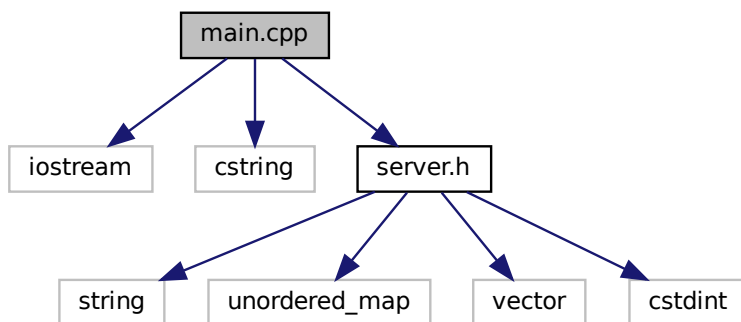
# Файлы

### 4.1 Файл main.cpp

Точка входа серверного приложения.

```
#include <iostream>
#include <cstring>
#include "server.h"
```

Граф включаемых заголовочных файлов для main.cpp:



### Функции

- void `showHelp` ()  
Выводит справочную информацию о параметрах командной строки.
- int `main` (int argc, char \*argv[])  
Основная функция серверного приложения.

#### 4.1.1 Подробное описание

Точка входа серверного приложения.

Автор

Чернышев Ринат Рустямович

Дата

26.12.2025

Основная программа, которая парсит аргументы командной строки, создает и запускает экземпляр сервера для обработки клиентских подключений.

#### 4.1.2 Функции

##### 4.1.2.1 main()

```
int main (
    int argc,
    char * argv[] )
```

Основная функция серверного приложения.

Аргументы

argc	Количество аргументов командной строки.
argv	Массив строк-аргументов.

Возвращает

Код завершения программы:

- 0: успешное завершение или показ справки
- 1: ошибка запуска сервера

##### 4.1.2.2 showHelp()

```
void showHelp ( )
```

Выводит справочную информацию о параметрах командной строки.

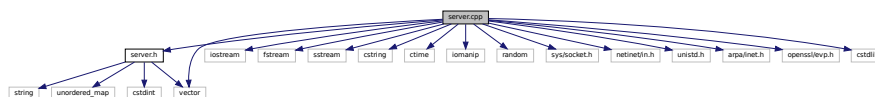
Отображает список доступных опций для запуска сервера и их значения по умолчанию.

## 4.2 Файл server.cpp

Реализация методов класса [Server](#) для обработки клиентских подключений.

```
#include "server.h"
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <cstring>
#include <ctime>
#include <iomanip>
#include <random>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <openssl/evp.h>
#include <cstdlib>
```

Граф включаемых заголовочных файлов для server.cpp:



### 4.2.1 Подробное описание

Реализация методов класса [Server](#) для обработки клиентских подключений.

Автор

Чернышев Ринат Рустямович

Дата

26.12.2025

Содержит реализацию сетевого сервера с аутентификацией пользователей по протоколу SHA-224, обработкой векторных данных и логированием событий.

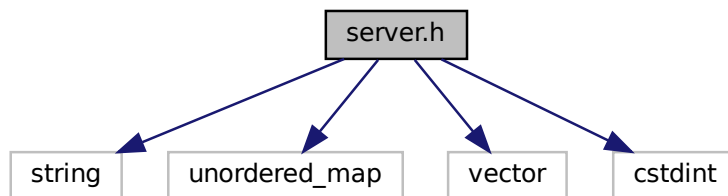
## 4.3 Файл server.h

Заголовочный файл класса [Server](#).

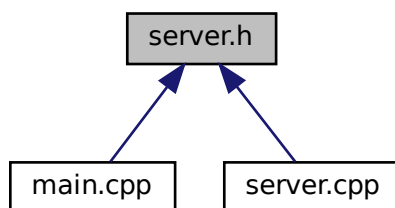
```
#include <string>
#include <unordered_map>
#include <vector>
```

```
#include <cstdint>
```

Граф включаемых заголовочных файлов для server.h:



Граф файлов, в которые включается этот файл:



## Классы

- class [Server](#)

Класс сервера для обработки клиентских подключений.

### 4.3.1 Подробное описание

Заголовочный файл класса [Server](#).

Автор

Чернышев Ринат Рустямович

Дата

26.12.2025

Объявление класса сервера для обработки сетевых подключений, аутентификации клиентов по протоколу SHA-224 и вычисления суммы квадратов векторов.

# Предметный указатель

- authenticate
  - Server, [6](#)
- calculateSumOfSquares
  - Server, [7](#)
- generateSalt
  - Server, [8](#)
- handleClient
  - Server, [8](#)
- loadUserDatabase
  - Server, [8](#)
- logError
  - Server, [9](#)
- logPath
  - Server, [11](#)
- main
  - main.cpp, [14](#)
- main.cpp, [13](#)
  - main, [14](#)
  - showHelp, [14](#)
- port
  - Server, [11](#)
- processVectors
  - Server, [9](#)
- readExact
  - Server, [10](#)
- Server, [5](#)
  - authenticate, [6](#)
  - calculateSumOfSquares, [7](#)
  - generateSalt, [8](#)
  - handleClient, [8](#)
  - loadUserDatabase, [8](#)
  - logError, [9](#)
  - logPath, [11](#)
  - port, [11](#)
  - processVectors, [9](#)
  - readExact, [10](#)
  - Server, [6](#)
  - sha224Hash, [10](#)
  - start, [11](#)
  - userDbPath, [11](#)
  - users, [12](#)
- server.cpp, [15](#)
- server.h, [15](#)
- sha224Hash
  - Server, [10](#)
- showHelp
  - main.cpp, [14](#)
- start
  - Server, [11](#)
- userDbPath
  - Server, [11](#)
- users
  - Server, [12](#)