Программная реализация сетевого сервера

1.0

Создано системой Doxygen 1.9.1

Глава 1

Иерархический список классов

1.1 Иерархия классов

Иерархия классов.

Глава 2

Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

Глава 3

Список файлов

3.1 Файлы

Полный список документированных файлов.

Глава 4

Классы

4.1 Класс Authorized

Класс для обработки авторизации клиентов.

#include <server.h>

Открытые члены

- Authorized (Error &err)
  Конструктор класса Authorized.
- void msgsend (int work_sock, const std::string &mess)
  Отправляет сообщение клиенту.
- std::string SHA (const std::string &sah)
  Вычисляет SHA-224 хеш строки.
- std::string salt_generator (const std::size_t length)
  Генерирует случайную соль заданной длины.
- int authorized (int work_sock, std::string fileName, std::string file_error)
  Проверяет авторизационные данные клиента.

4.1.1 Подробное описание

Класс для обработки авторизации клиентов.

Этот класс отвечает за проверку авторизационных данных клиента, включая генерацию соли и хеширование паролей.

4.1.2 Конструктор(ы)

4.1.2.1 Authorized()

```
Authorized::Authorized (
            Error & err )   [inline]
```

Конструктор класса Authorized.

Этот конструктор инициализирует объект класса Authorized, принимая ссылку на объект класса Error. Объект класса Error используется для обработки и записи ошибок, возникающих в процессе авторизации клиентов.

Аргументы

err	Ссылка на объект класса Error для обработки ошибок.

4.1.3 Методы

4.1.3.1 authorized()

```
int Authorized::authorized (
            int work_sock,
            std::string fileName,
            std::string file_error )
```

Проверяет авторизационные данные клиента.

Эта функция проверяет авторизационные данные клиента, сравнивая их с данными, хранящимися в указанном файле. Если авторизация проходит успешно, функция возвращает 1. В случае ошибки возвращается -1, и ошибка записывается в указанный файл.

Аргументы

work_sock	Дескриптор сокета для работы с клиентом.
fileName	Имя файла с авторизационными данными.
file_error	Имя файла для записи ошибок.

Возвращает

1, если авторизация прошла успешно, иначе -1.

Исключения

RuntimeError	Если произошла ошибка при проверке авторизации.

4.1.3.2 msgsend()

```
void Authorized::msgsend (
            int work_sock,
            const std::string & mess )
```

Отправляет сообщение клиенту.

Эта функция отправляет строковое сообщение клиенту через указанный дескриптор сокета. Если при отправке сообщения возникает ошибка, функция записывает сообщение об ошибке и выбрасывает исключение.

Аргументы

work_sock	Дескриптор сокета для работы с клиентом.
mess	Сообщение для отправки.

Исключения

RuntimeError	Если произошла ошибка при отправке сообщения.

4.1.3.3 salt_generator()

```
std::string Authorized::salt_generator (
            const std::size_t length )
```

Генерирует случайную соль заданной длины.

Эта функция генерирует случайную строку заданной длины, состоящую из символов, выбранных из заданного набора символов. Если при генерации строки возникает ошибка, функция записывает сообщение об ошибке и возвращает пустую строку.

Аргументы

length	Длина генерируемой строки.

Возвращает

Сгенерированная соль.

Исключения

RuntimeError	Если произошла ошибка при генерации соли.

4.1.3.4 SHA()

```
std::string Authorized::SHA (
            const std::string & sah )
```

Вычисляет SHA-224 хеш строки.

Эта функция вычисляет SHA-224 хеш для переданной строки и возвращает результат в виде шестнадцатеричной строки. Если при вычислении хеша возникает ошибка, функция записывает сообщение об ошибке и возвращает пустую строку.

Аргументы

sah	Строка для хеширования.

Возвращает

Хеш строки в виде шестнадцатеричной строки.

Исключения

RuntimeError	Если произошла ошибка при вычислении хеша.

Объявления и описания членов классов находятся в файлах:

- server.h
- authorization.cpp

4.2 Класс Calculator

Класс для выполнения вычислений над векторами.

#include <server.h>

Открытые статические члены

- static double processVector (const std::vector< double > &vectorValues)
  Вычисляет произведение элементов вектора.

4.2.1 Подробное описание

Класс для выполнения вычислений над векторами.

Этот класс вычисления произведение элементов вектора.

4.2.2 Методы

4.2.2.1 processVector()

```
double Calculator::processVector (
            const std::vector< double > & vectorValues )   [static]
```

Вычисляет произведение элементов вектора.

Эта статическая функция вычисляет произведение всех элементов в переданном векторе значений типа double. Если вектор пуст, функция возвращает 1.0, так как произведение элементов пустого вектора считается равным 1.

Аргументы

vectorValues	Вектор значений для вычисления произведения.

Возвращает

Произведение элементов вектора.

Объявления и описания членов классов находятся в файлах:

- server.h
- calculator.cpp

4.3 Класс Error

Класс для обработки и записи ошибок в файл.

#include <server.h>

Открытые члены

- Error ()
  -Конструктор по умолчанию

Открытые статические члены

- static void errors (std::string error, std::string name)
  Записывает ошибку в файл.
- static int er (std::string fileName, std::string file_error)
  Проверяет доступность файла и записывает ошибку, если файл недоступен.

4.3.1 Подробное описание

Класс для обработки и записи ошибок в файл.

Этот класс предоставляет методы для записи ошибок в файл и проверки доступности файлов.

4.3.2 Методы

4.3.2.1 er()

```
int Error::er (
            std::string fileName,
            std::string file_error )   [static]
```

Проверяет доступность файла и записывает ошибку, если файл недоступен.

Эта статическая функция проверяет, доступен ли указанный файл для чтения. Если файл недоступен, функция записывает сообщение об ошибке в указанный файл ошибок.

Аргументы

fileName	Имя файла для проверки.
file_error	Имя файла для записи ошибок.

Возвращает

1, если файл доступен, иначе выбрасывает исключение.

Исключения

RuntimeError	Если файл недоступен.

4.3.2.2 errors()

```
void Error::errors (
            std::string error,
            std::string name )   [static]
```

Записывает ошибку в файл.

Эта статическая функция записывает переданную строку ошибки в указанный файл. Функция также добавляет текущую дату и время к записи ошибки.

Аргументы

error	Текст ошибки.
name	Имя файла для записи ошибки.

Объявления и описания членов классов находятся в файлах:

- server.h
- error.cpp
- server.cpp

4.4 Класс Interface

Класс для обработки интерфейса командной строки.

#include <server.h>

Открытые члены

- Interface ()

  Конструктор класса Interface.
- std::string getFileName () const

  Возвращает имя файла с базой данных пользователей.
- int getPort () const

  Возвращает номер порта для сервера.
- std::string getErrorFile () const

  Возвращает имя файла для записи ошибок.
- std::string getDescription () const

  Возвращает описание команд и параметров.
- bool parseArguments (int argc, char *argv[])

  Разбирает аргументы командной строки.

4.4.1 Подробное описание

Класс для обработки интерфейса командной строки.

Этот класс отвечает за разбор аргументов командной строки и предоставление доступа к параметрам.

4.4.2 Конструктор(ы)

4.4.2.1 Interface()

Interface::Interface ( ) [inline]

Конструктор класса Interface.

Этот конструктор инициализирует объект класса Interface, устанавливая значения по умолчанию для его членов. Конструктор использует список инициализации для установки начальных значений для переменных fileName, port, file_error и description.

Инициализирует параметры по умолчанию:

- fileName: "base.txt"
- port: 33333
- file_error: "error.txt"
- description: "" (пустая строка)

4.4.3 Методы

4.4.3.1 getDescription()

std::string Interface::getDescription ( ) const

Возвращает описание команд и параметров.

Эта функция возвращает значение description, которое содержит описание команд и параметров, используемых в интерфейсе командной строки. Функция является константной, что означает, что она не изменяет состояние объекта.

Возвращает

Описание команд и параметров.

4.4.3.2 getErrorFile()

std::string Interface::getErrorFile ( ) const

Возвращает имя файла для записи ошибок.

Эта функция возвращает значение file_error, которое содержит имя файла, в который будут записываться ошибки. Функция является константной, что означает, что она не изменяет состояние объекта.

Возвращает

Имя файла для записи ошибок.

4.4.3.3 getFileName()

std::string Interface::getFileName ( ) const

Возвращает имя файла с базой данных пользователей.

Эта функция возвращает значение fileName, которое содержит имя файла с базой данных пользователей. Функция является константной, что означает, что она не изменяет состояние объекта.

Возвращает

Имя файла с базой данных пользователей.

4.4.3.4 getPort()

```
int Interface::getPort ( ) const
```

Возвращает номер порта для сервера.

Эта функция возвращает значение port, которое содержит номер порта, на котором сервер будет прослушивать подключения. Функция является константной, что означает, что она не изменяет состояние объекта.

Возвращает

Номер порта для сервера.

4.4.3.5 parseArguments()

```
bool Interface::parseArguments (
            int argc,
            char * argv[] )
```

Разбирает аргументы командной строки.

Эта функция разбирает аргументы командной строки, переданные при запуске программы. Она обрабатывает различные опции, такие как --help, --file, --port, и --error, и устанавливает соответствующие значения. Если разбор аргументов прошел успешно, функция возвращает true. В случае ошибки выбрасывается исключение.

Аргументы

argc	Количество аргументов.
argv	Массив аргументов.

Возвращает

true, если разбор прошел успешно, иначе выбрасывает исключение.

Исключения

RuntimeError	Если произошла ошибка при разборе аргументов.

Объявления и описания членов классов находятся в файлах:

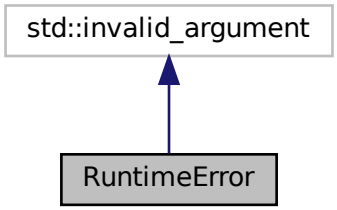
- server.h
- Interface.cpp

4.5 Класс RuntimeError

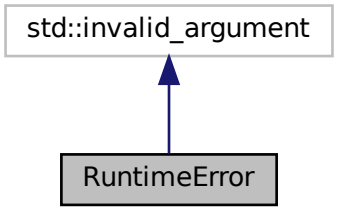
Класс для обработки исключений.

#include <server.h>

Граф наследования:RuntimeError:



Граф связей класса RuntimeError:



Открытые члены

- RuntimeError (const std::string &what_arg)
  Конструктор класса RuntimeError.
- RuntimeError (const char *what_arg)
  Конструктор класса RuntimeError.

4.5.1 Подробное описание

Класс для обработки исключений.

Этот класс наследуется от std::invalid_argument и используется для выбрасывания исключений с сообщениями об ошибках.

4.5.2 Конструктор(ы)

4.5.2.1 RuntimeError() [1/2]

RuntimeError::RuntimeError (
const std::string & what_arg) [inline], [explicit]

Конструктор класса RuntimeError.

Этот конструктор инициализирует объект класса RuntimeError, принимая строку с сообщением об ошибке. Объект класса RuntimeError наследуется от std::invalid_argument, что позволяет использовать его как исключение.

Аргументы

const	const std::string& what_arg Сообщение об ошибке.

4.5.2.2 RuntimeError() [2/2]

RuntimeError::RuntimeError (
const char * what_arg) [inline], [explicit]

Конструктор класса RuntimeError.

Этот конструктор инициализирует объект класса RuntimeError, принимая строку с сообщением об ошибке. Объект класса RuntimeError наследуется от std::invalid_argument, что позволяет использовать его как исключение.

Аргументы

const	const char* what_arg Сообщение об ошибке.

Объявления и описания членов класса находятся в файле:

- server.h

4.6 Класс Server

Класс для управления сервером и обработки подключений клиентов.

#include <server.h>

Открытые члены

- Server (Error &err)
  Конструктор класса Server.
- int self_addr (std::string &error, std::string &file_error, int port)
  Настраивает сервер для прослушивания заданного порта.
- int client_addr (int s, std::string &error, std::string &file_error)
  Ожидает и принимает подключение клиента.

4.6.1 Подробное описание

Класс для управления сервером и обработки подключений клиентов.

Этот класс отвечает за настройку сервера, прослушивание порта и обработку подключений клиентов.

4.6.2 Конструктор(ы)

4.6.2.1 Server()

```
Server::Server (
            Error & err )   [inline]
```

Конструктор класса Server.

Этот конструктор инициализирует объект класса Server, принимая ссылку на объект класса Error. Объект класса Error используется для обработки и записи ошибок, возникающих в процессе работы сервера.

Аргументы

err	Ссылка на объект класса Error для обработки ошибок.

4.6.3 Методы

4.6.3.1 client_addr()

```
int Server::client_addr (
            int s,
            std::string & error,
            std::string & file_error )
```

Ожидает и принимает подключение клиента.

Эта функция ожидает подключения клиента на указанном дескрипторе сокета. Когда клиент подключается, функция возвращает дескриптор сокета для работы с клиентом. Если подключение не удается, функция возвращает -1 и записывает ошибку в указанный файл.

Аргументы

s	Дескриптор сокета, на котором сервер прослушивает подключения.
error	Ссылка на строку для записи ошибок.
file_error	Имя файла для записи ошибок.

Возвращает

Дескриптор сокета для работы с клиентом, если подключение успешно, иначе -1.

Исключения

RuntimeError	Если произошла ошибка при подключении клиента.

4.6.3.2 self_addr()

```
int Server::self_addr (
                std::string & error,
                std::string & file_error,
                int port )
```

Настраивает сервер для прослушивания заданного порта.

Эта функция выполняет настройку сервера для прослушивания подключений на указанном порту. Она создает сокет, устанавливает параметры сокета, настраивает таймауты и привязывает сокет к указанному порту. Если настройка прошла успешно, функция возвращает дескриптор сокета. В случае ошибки возвращается -1, и ошибка записывается в указанный файл.

Аргументы

error	Ссылка на строку для записи ошибок.
file_error	Имя файла для записи ошибок.
port	Порт, на котором сервер будет прослушивать подключения.

Возвращает

Дескриптор сокета, если настройка прошла успешно, иначе -1.

Исключения

RuntimeError	Если произошла ошибка при настройке сервера.

Объявления и описания членов классов находятся в файлах:

- server.h
- server.cpp

Глава 5

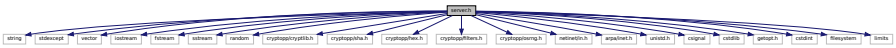
Файлы

5.1 Файл server.h

Основной заголовочный файл для сервера.

```
#include <string>
#include <stdexcept>
#include <vector>
#include <iostream>
#include <fstream>
#include <sstream>
#include <random>
#include <cryptopp/cryptlib.h>
#include <cryptopp/sha.h>
#include <cryptopp/hex.h>
#include <cryptopp/filters.h>
#include <cryptopp/osrng.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <csignal>
#include <cstdlib>
#include <getopt.h>
#include <cstdint>
#include <filesystem>
#include <limits>
```

Граф включаемых заголовочных файлов для server.h:



Классы

- class Error

  Класс для обработки и записи ошибок в файл.
- class Server

  Класс для управления сервером и обработки подключений клиентов.

- class Authorized

  Класс для обработки авторизации клиентов.
- class Calculator

  Класс для выполнения вычислений над векторами.
- class RuntimeError

  Класс для обработки исключений.
- class Interface

  Класс для обработки интерфейса командной строки.

5.1.1 Подробное описание

Основной заголовочный файл для сервера.

Автор

Синельникова Т.А.

Версия

1.0

Дата

05.12.2024

Авторство

ИБСТ ПГУ

Этот файл содержит определения классов и функций, необходимых для работы сервера, включая обработку ошибок, авторизацию, вычисления и интерфейс командной строки.

Предметный указатель