Сетевой сервер.

Создано системой Doxygen 1.9.4

Глава 1

Алфавитный указатель классов

1.1 Классы

Классы с их кратким описанием.

Глава 2

Список файлов

2.1 Файлы

Полный список документированных файлов.

Глава 3

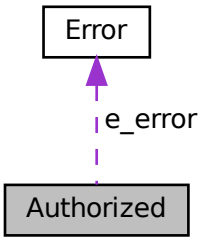
Классы

3.1 Класс Authorized

Класс для авторизации клиента.

#include <calc.h>

Граф связей класса Authorized:



Открытые члены

- std::string MD (const std::string &sah)
  Метод для вычисления MD5 хеша.
- Authorized (Error err)
  Конструктор класса Authorized.
- int authorized (int work_sock, string file_name, string file_error)
  Метод для авторизации клиента.

Закрытые члены

- void msgsend (int work_sock, const std::string &mess)
  Метод для отправки сообщения клиенту.

Закрытые данные

- Error e_error
  Объект для обработки ошибок.

3.1.1 Подробное описание

Класс для авторизации клиента.

3.1.2 Конструктор(ы)

3.1.2.1 Authorized()

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

Конструктор класса Authorized.

Аргументы

err	Объект класса Error для обработки ошибок.

3.1.3 Методы

3.1.3.1 authorized()

```
int Authorized::authorized (
                int work_sock,
                string file_name,
                string file_error )
```

Метод для авторизации клиента.

Аргументы

work_sock	Сокет клиента.
file_name	Имя файла с данными для авторизации.
file_error	Имя файла для записи ошибки.

Возвращает

Результат авторизации (0 - успех, 1 - ошибка).

Метод принимает сообщение от клиента через сокет work_sock. Это сообщение содержит данные для авторизации (логин, соль и хеш пароля).

```
recv(work_sock, &msg, sizeof(msg), 0);
string message = msg;
```

Метод открывает файл file_name, который содержит данные авторизации (логин и хеш пароля).

```
fstream file;
file.open(file_name);
getline(file, login, ':');
getline(file, hashq);
```

Метод сравнивает логин, полученный от клиента, с логином, хранящимся в файле. Если логины не совпадают, метод отправляет клиенту сообщение ERR, записывает сообщение об ошибке в файл (через объект e_error) и закрывает соединение.

```
if (message.substr(0, login.length()) != login) {
    msgsend(work_sock, err);
    error = "Ошибка логина";
    e_error.errors(error, file_error);
    close(work_sock);
    return 1;
}
```

Если логин корректен, метод извлекает соль из сообщения клиента (16 символов после логина). Затем метод формирует строку sah, которая состоит из соли и хеша пароля из файла. Далее метод вычисляет MD5-хеш строки sah с помощью функции MD.

```
string salt = message.substr(login.length(), 16); // Извлекаем соль из сообщения
string sah = salt + hashq;
string digest = MD(sah);
```

Метод сравнивает вычисленный хеш digest с хешем, полученным от клиента. Если хеши не совпадают, метод отправляет клиенту сообщение ERR, записывает сообщение об ошибке в файл и закрывает соединение.

```
string received_hash = message.substr(login.length() + 16);
if (digest != received_hash) {
    msgsend(work_sock, err);
    error = "Ошибка пароля";
    e_error.errors(error, file_error);
    close(work_sock);
    return 1;
}
```

Если логин и пароль корректны, метод отправляет клиенту сообщение OK и возвращает 0, что означает успешную авторизацию.

3.1.3.2 MD()

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

Метод для вычисления MD5 хеша.

Аргументы

sah	Строка для хеширования.

Возвращает

Хеш в формате строки.

3.1.3.3 msgsend()

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

Метод для отправки сообщения клиенту.

Аргументы

work_sock	Сокет клиента.
mess	Сообщение для отправки.

Объявления и описания членов классов находятся в файлах:

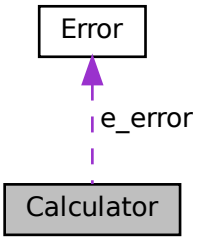
- calc.h
- calc.cpp

3.2 Класс Calculator

Класс для выполнения вычислений.

#include <calc.h>

Граф связей класса Calculator:



Открытые члены

- Calculator (Error err)

  Конструктор класса Calculator.
- int calc (int work_sock)

  Метод для выполнения вычислений.

Закрытые данные

- Error e_error

3.2.1 Подробное описание

Класс для выполнения вычислений.

3.2.2 Конструктор(ы)

3.2.2.1 Calculator()

```
Calculator::Calculator (
            Error err )   [inline]
```

Конструктор класса Calculator.

Аргументы

err	Объект класса Error для обработки ошибок.

3.2.3 Методы

3.2.3.1 calc()

```
int Calculator::calc (
            int work_sock )
```

Метод для выполнения вычислений.

Аргументы

work_sock	Сокет клиента.

Возвращает

Результат вычислений (1 - успех).

Метод принимает от клиента значение Quantity, которое указывает, сколько наборов чисел будет отправлено.
recv(work_sock, &Quantity, sizeof(Quantity), 0);

Метод выполняет цикл по количеству наборов чисел (Quantity). Для каждого набора: Принимает значение Length, которое указывает, сколько чисел содержится в текущем наборе. Инициализирует переменную Amount для хранения суммы чисел в текущем наборе.

```
for (int j = 0; j < Quantity; j++) {
    recv(work_sock, &Length, sizeof(Length), 0);
    float Amount = 0;
```

Внутри цикла по Length метод принимает числа от клиента и суммирует их в переменной Amount.

```
for (int i = 0; i < Length; i++) {
    recv(work_sock, &Vector_numbers, sizeof(Vector_numbers), 0);
    Amount += Vector_numbers;
}
```

После суммирования всех чисел в наборе метод отправляет результат (Amount) обратно клиенту.

```
send(work_sock, &Amount, sizeof(Amount), 0);
```

После завершения всех вычислений метод выводит закрывает соединение с клиентом.

Объявления и описания членов классов находятся в файлах:

- calc.h
- calc.cpp

3.3 Класс Error

Класс для обработки и записи ошибок

#include <calc.h>

Открытые члены

- Error ()
  Конструктор по умолчанию

Открытые статические члены

- static void errors (std::string error, std::string name)
  Метод для записи ошибки в файл.
- static int er (std::string file_name, std::string file_error)
  Метод для проверки доступности файла.

3.3.1 Подробное описание

Класс для обработки и записи ошибок

3.3.2 Методы

3.3.2.1 er()

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

Метод для проверки доступности файла.

Аргументы

file_name	Имя файла с базой данных.
file_error	Имя файла для записи ошибки.

Возвращает

Код ошибки (12, если файл недоступен).

3.3.2.2 errors()

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

Метод для записи ошибки в файл.

Аргументы

error	Текст ошибки.
name	Имя файла для записи ошибки.

Объявления и описания членов классов находятся в файлах:

- calc.h
- calc.cpp

3.4 Класс interface

Класс для обработки аргументов командной строки.

#include <interface.h>

Открытые члены

- interface (int argc, char *argv[])
  Конструктор класса interface.
- int arguments ()
  Метод для обработки аргументов командной строки.
- std::string get_file_name () const
  Получить имя файла базы данных.
- int get_port () const
  Получить номер порта.
- std::string get_file_error () const
  Получить имя файла для записи ошибок.

Закрытые члены

- void print_help () const
  Метод для вывода справки.

Закрытые данные

- int argc_
  < Количество аргументов командной строки.
- char ** argv_
  < Массив аргументов командной строки..
- std::string file_name
  < Имя файла с базой данных.
- int port
  < Номер порта.
- std::string file_error
  < Имя файла для записи ошибок.

3.4.1 Подробное описание

Класс для обработки аргументов командной строки.

3.4.2 Конструктор(ы)

3.4.2.1 interface()

```
interface::interface (
                int argc,
                char * argv[] )
```

Конструктор класса interface.

Аргументы

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

3.4.3 Методы

3.4.3.1 arguments()

```
int interface::arguments ( )
```

Метод для обработки аргументов командной строки.

Возвращает

0, если аргументы обработаны успешно, иначе 1.

3.4.3.2 get_file_error()

std::string interface::get_file_error ( ) const

Получить имя файла для записи ошибок.

Возвращает

Имя файла для записи ошибок.

3.4.3.3 get_file_name()

std::string interface::get_file_name ( ) const

Получить имя файла базы данных.

Возвращает

Имя файла.

3.4.3.4 get_port()

int interface::get_port ( ) const

Получить номер порта.

Возвращает

Номер порта.

Объявления и описания членов классов находятся в файлах:

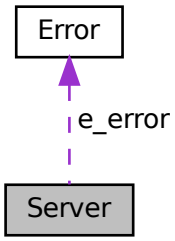
- interface.h
- interface.cpp

3.5 Класс Server

Класс для настройки и управления сервером.

#include <calc.h>

Граф связей класса Server:



Открытые члены

- Server (Error err)
  Конструктор класса Server.
- int self_addr (std::string &error, std::string &file_error, int port)
  Метод для настройки адреса сервера.
- int client_addr (int s, std::string &error, std::string &file_error)
  Метод для обработки подключения клиента.

Закрытые данные

- Error e_error
  Объект для обработки ошибок.

3.5.1 Подробное описание

Класс для настройки и управления сервером.

3.5.2 Конструктор(ы)

3.5.2.1 Server()

Server::Server (
Error err) [inline]

Конструктор класса Server.

Аргументы

err	Объект класса Error для обработки ошибок.

3.5.3 Методы

3.5.3.1 client_addr()

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

Метод для обработки подключения клиента.

Аргументы

s	Сокет сервера.
error	Ссылка на строку для записи ошибки.
file_error	Имя файла для записи ошибки.

Возвращает

Сокет клиента.

3.5.3.2 self_addr()

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

Метод для настройки адреса сервера.

Аргументы

error	Ссылка на строку для записи ошибки.
file_error	Имя файла для записи ошибки.
port	Порт сервера.

Возвращает

Сокет сервера.

Метод начинает с проверки, является ли переданный порт корректным. Порт должен быть в диапазоне от 0 до 65535. Если порт выходит за пределы допустимого диапазона, метод формирует сообщение об ошибке, записывает его в файл (через объект e_error), и выбрасывает исключение std::runtime_error.

```
if (port < 0 || port > 65535) {
    error = "Некорректное значение порта: " + to_string(port);
    e_error.errors(error, file_error);
    throw std::runtime_error("Invalid port value");
    return -1;
}
```

Метод создает сокет с помощью функции socket. Сокет использует семейство адресов AF_INET (IPv4) и тип сокета SOCK_STREAM (TCP). Если создание сокета завершается с ошибкой (возвращается значение меньше 0), метод выводит сообщение об ошибке с помощью perror и завершает программу с кодом EXIT_FAILURE.

```
int sock = socket(AF_INET, SOCK_STREAM, 0);
if (sock < 0) {
    perror("Ошибка при создании сокета");
    exit(EXIT_FAILURE);
}
```

Метод устанавливает опцию SO_REUSEADDR для сокета, чтобы разрешить повторное использование адреса и порта после завершения работы сервера. Если установка опции завершается с ошибкой (возвращается -1), метод выбрасывает исключение std::system_error.

```
int on = 1;
int rc = setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on));
if (rc == -1) {
    throw system_error(errno, generic_category(), "Ошибка установки сокета");
}
```

Метод устанавливает обработчик сигнала SIGALRM с помощью функции signal. Этот сигнал будет использоваться для ограничения времени ожидания подключения клиента. Затем метод устанавливает таймаут на 240 секунд для операций приема данных с помощью опции SO_RCVTIMEO. Если установка таймаута завершается с ошибкой, метод выбрасывает исключение std::system_error.

```
signal(SIGALRM, alarm_handler);
alarm(240);
struct timeval timeout {240, 0};
rc = setsockopt(sock, SOL_SOCKET, SO_RCVTIMEO, &timeout, sizeof(timeout));
if (rc == -1) {
    throw system_error(errno, generic_category(), "Ошибка установки сокета");
}
```

Метод настраивает структуру sockaddr_in, которая содержит информацию об адресе сервера.

```
sockaddr_in self_addr;
self_addr.sin_family = AF_INET;
self_addr.sin_port = htons(port);
self_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

Метод вызывает функцию bind, чтобы привязать сокет к указанному адресу и порту. Если привязка завершается с ошибкой (возвращается -1), метод выводит сообщение об ошибке, записывает его в файл (через объект e_error) и возвращает код ошибки 1.

```
int b = bind(sock, reinterpret_cast<sockaddr*>(&self_addr), sizeof(self_addr));
if (b == -1) {
    cout « "Ошибка привязки\n";
    error = "Ошибка";
    e_error.errors(error, file_error);
    return 1;
}
```

После успешной привязки сокета метод вызывает функцию listen, чтобы начать прослушивание входящих подключений.

Объявления и описания членов классов находятся в файлах:

- calc.h
- calc.cpp

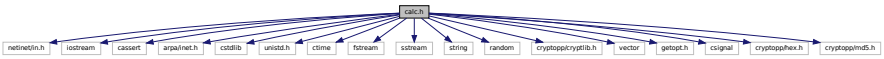
Глава 4

Файлы

4.1 Файл calc.h

Заголовочный файл для модуля calc.

#include <netinet/in.h>
#include <iostream>
#include <cassert>
#include <arpa/inet.h>
#include <cstdlib>
#include <unistd.h>
#include <ctime>
#include <fstream>
#include <sstream>
#include <string>
#include <random>
#include <cryptopp/cryptlib.h>
#include <vector>
#include <getopt.h>
#include <csignal>
#include <cryptopp/hex.h>
#include <cryptopp/md5.h>
Граф включаемых заголовочных файлов для calc.h:



Классы

- class Error

  Класс для обработки и записи ошибок
- class Server

  Класс для настройки и управления сервером.
- class Authorized

  Класс для авторизации клиента.
- class Calculator

  Класс для выполнения вычислений.

Макросы

- #define CRYPTOPP_ENABLE_NAMESPACE_WEAK 1

4.1.1 Подробное описание

Заголовочный файл для модуля calc.

Автор

Кулагина П.В.

Версия

1.0

Дата

23.12.2024

Авторство

ИБСТ ПГУ

4.2 calc.h

См. документацию.

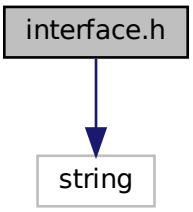
```
1
8 #include <netinet/in.h>
9 #include <iostream>
10 #include <cassert>
11 #include <arpa/inet.h>
12 #include <cstdlib>
13 #include <unistd.h>
14 #include <ctime>
15 #include <fstream>
16 #include <sstream>
17 #include <string>
18 #include <random>
19 #include <cryptopp/cryptlib.h>
20 #include <iostream>
21 #include <vector>
22 #include <getopt.h>
23 #include <unistd.h>
24 #include <getopt.h>
25 #include <csignal>
26 #include <cstdlib>
27 #include <cryptopp/hex.h>
28 #define CRYPTOPP_ENABLE_NAMESPACE_WEAK 1
29 #include <cryptopp/md5.h>
30 using namespace CryptoPP;
31 using namespace std;
32
36 class Error
37 {
38 public:
39     Error();
45     static void errors(std::string error, std::string name);
52     static int er(std::string file_name, std::string file_error);
53 };
54
59 class Server
60 {
61 private:
```

```
62     Error e_error;
63 public:
68     Server(Error err) : e_error(err) {
69         e_error = err;
70     }
131     int self_addr(std::string& error, std::string& file_error, int port);
139     int client_addr(int s, std::string& error, std::string& file_error);
140 };
141
142
147 class Authorized
148 {
149 private:
150     Error e_error;
156     void msgsend(int work_sock, const std::string& mess);
157 public:
163     std::string MD(const std::string& sah);
168     Authorized(Error err) : e_error(err) {
169         e_error = err;
170     }
218     int authorized(int work_sock, string file_name, string file_error);
219 };
220
221
226 class Calculator
227 {
228 private:
229     Error e_error; //< Объект для обработки ошибок.
230 public:
235     Calculator(Error err) : e_error(err) {
236         e_error = err;
237     }
265     int calc(int work_sock);
266 };
267
```

4.3 Файл interface.h

Заголовочный файл для класса interface.

#include <string>
Граф включаемых заголовочных файлов для interface.h:



Классы

- class interface

  Класс для обработки аргументов командной строки.

4.3.1 Подробное описание

Заголовочный файл для класса interface.

4.4 interface.h

См. документацию.

```
1
5 #ifndef INTERFACE_H
6 #define INTERFACE_H
7 #include <string>
8
13 class interface {
14 public:
20     interface(int argc, char *argv[]);
25     int arguments();
30     std::string get_file_name() const;
35     int get_port() const;
40     std::string get_file_error() const;
41 private:
45     void print_help() const;
46     int argc_;
47     char **argv_;
48     std::string file_name;
49     int port;
50     std::string file_error;
51 };
52 #endif
```

Предметный указатель

client_addr, 15
self_addr, 15
Server, 14