Server-MD5-int32_t-saltOnClient

Создано системой Doxygen 1.9.4

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 CLI

Класс для обработки аргументов командной строки.

#include <cli.h>

Открытые члены

- CLI ()
  Конструктор по умолчанию.
- void parseArgs (int argc, char *argv[])
  Парсит аргументы командной строки.
- void showHelp () const
  Печатает справку по использованию программы.
- string getAddr () const
  Возвращает адрес сервера.
- int getPort () const
  Возвращает порт сервера.
- string getLogFile () const
  Возвращает путь к лог-файлу.
- string getDataFile () const
  Возвращает путь к файлу данных.

4.1.1 Подробное описание

Класс для обработки аргументов командной строки.

Класс предоставляет методы для парсинга аргументов командной строки и получения значений аргументов.

4.1.2 Конструктор(ы)

4.1.2.1 CLI()

CLI::CLI ( )

Конструктор по умолчанию.

Устанавливает значения аргументов командной строки по умолчанию.

4.1.3 Методы

4.1.3.1 getAddr()

string CLI::getAddr ( ) const

Возвращает адрес сервера.

Возвращает

string Адрес сервера.

4.1.3.2 getDataFile()

string CLI::getDataFile ( ) const

Возвращает путь к файлу данных.

Возвращает

string Путь к файлу данных.

4.1.3.3 getLogFile()

string CLI::getLogFile ( ) const

Возвращает путь к лог-файлу.

Возвращает

string Путь к лог-файлу.

4.1.3.4 getPort()

int CLI::getPort ( ) const

Возвращает порт сервера.

Возвращает

int Порт сервера.

4.1.3.5 parseArgs()

void CLI::parseArgs (
int argc,
char * argv[])

Парсит аргументы командной строки.

Аргументы

argc	Количество аргументов.
argv	Массив аргументов.

Исключения

Error	Если возникает ошибка парсинга.

Объявления и описания членов классов находятся в файлах:

- code/cli.h
- code/cli.cpp

4.2 Класс DataWorker

Класс для работы с данными из файла.

#include <dataworker.h>

Открытые члены

- DataWorker (const string &filePath)
  Конструктор класса DataWorker.
- vector< pair< string, string > > readCredentials () const
  Читает учетные данные из файла.
- string getFilePath () const
  Возвращает путь к файлу данных.

4.2.1 Подробное описание

Класс для работы с данными из файла.

Класс предоставляет методы для чтения учетных данных из файла.

4.2.2 Конструктор(ы)

4.2.2.1 DataWorker()

DataWorker::DataWorker (
const string & filePath) [explicit]

Конструктор класса DataWorker.

Аргументы

filePath	Путь к файлу данных.

4.2.3 Методы

4.2.3.1 getFilePath()

string DataWorker::getFilePath ( ) const

Возвращает путь к файлу данных.

Возвращает

string Путь к файлу данных.

4.2.3.2 readCredentials()

vector< pair< string, string > > DataWorker::readCredentials ( ) const

Читает учетные данные из файла.

Возвращает

vector<pair<string, string>> Вектор пар логин/пароль.

Исключения

Error	Если не удается открыть файл или данные имеют неправильный формат.

Объявления и описания членов классов находятся в файлах:

- code/dataworker.h
- code/dataworker.cpp

4.3 Класс Error

Класс для представления ошибок в приложении.

#include <error.h>

Граф наследования:Error:

4.4 Класс Logger

Класс для записи сообщений об ошибках в лог-файл.

#include <logger.h>

Открытые члены

- Logger (const string &logFilePath)
  Конструктор класса Logger.
- void write (const Error &exception)
  Записывает сообщение об ошибке в лог-файл.
- string getLogFilePath () const
  Возвращает путь к лог-файлу.

4.4.1 Подробное описание

Класс для записи сообщений об ошибках в лог-файл.

Класс предоставляет методы для записи сообщений об ошибках в лог-файл.

4.4.2 Конструктор(ы)

4.4.2.1 Logger()

Logger::Logger (
const string & logFilePath) [explicit]

Конструктор класса Logger.

Аргументы

logFilePath	Путь к лог-файлу.

Исключения

runtime_error	Если не удается открыть лог-файл.

4.4.3 Методы

4.4.3.1 getLogFilePath()

```
string Logger::getLogFilePath ( ) const
```

Возвращает путь к лог-файлу.

Возвращает

string Путь к лог-файлу.

4.4.3.2 write()

```
void Logger::write (
            const Error & exception )
```

Записывает сообщение об ошибке в лог-файл.

Аргументы

exception	Объект ошибки для записи.

Исключения

runtime_error	Если лог-файл не открыт.

Объявления и описания членов классов находятся в файлах:

- code/logger.h
- code/logger.cpp

4.5 Класс Server

Класс для реализации сервера.

#include <server.h>

Открытые члены

- Server (const std::string &addr, int port, const std::vector< std::pair< std::string, std::string > > &data)
  Конструктор класса Server.
- void initializeServer ()
  Инициализирует сервер.
- void terminateServer ()
  Завершает работу сервера.
- std::string & getAddr ()
  Возвращает адрес сервера.
- int & getPort ()
  Возвращает порт сервера.
- std::vector< std::pair< std::string, std::string > > & getData ()
  Возвращает учетные данные для аутентификации.
- void waitForClientConnection ()
  Ожидает подключения клиента.
- void authenticateClient ()
  Аутентифицирует клиента.
- void calculateAndSendProduct ()
  Вычисляет произведения значений в векторе и отправляет их клиенту.

4.5.1 Подробное описание

Класс для реализации сервера.

Класс предоставляет методы для инициализации сервера, ожидания подключения клиентов, аутентификации клиентов и вычисления произведений значений.

4.5.2 Конструктор(ы)

4.5.2.1 Server()

```
Server::Server (
            const std::string & addr,
            int port,
            const std::vector< std::pair< std::string, std::string > > & data )
```

Конструктор класса Server.

Аргументы

addr	Адрес сервера.
port	Порт сервера.
data	Вектор пар логин/пароль для аутентификации.

4.5.3 Методы

4.5.3.1 authenticateClient()

void Server::authenticateClient ( )

Аутентифицирует клиента.

Проверяет логин и пароль клиента, используя хеширование MD5.

Исключения

Error	Если аутентификация неуспешна.

4.5.3.2 calculateAndSendProduct()

void Server::calculateAndSendProduct ( )

Вычисляет произведения значений в векторе и отправляет их клиенту.

Исключения

Error	Если возникает ошибка при чтении или отправке данных.

4.5.3.3 getAddr()

string & Server::getAddr ( )

Возвращает адрес сервера.

Возвращает

std::string Адрес сервера.

4.5.3.4 getData()

vector< pair< string, string > > & Server::getData ( )

Возвращает учетные данные для аутентификации.

Возвращает

std::vector<std::pair<std::string, std::string>> Вектор пар логин/пароль.

4.5.3.5 getPort()

int & Server::getPort ( )

Возвращает порт сервера.

Возвращает

int Порт сервера.

4.5.3.6 initializeServer()

void Server::initializeServer ( )

Инициализирует сервер.

Настраивает серверное соединение и начинает прослушивание порта.

Исключения

Error	Если возникает ошибка при создании, привязке или прослушивании сокета.

4.5.3.7 terminateServer()

void Server::terminateServer ( )

Завершает работу сервера.

Закрывает все открытые соединения.

Исключения

Error	Если возникает ошибка при закрытии сокетов.

4.5.3.8 waitForClientConnection()

void Server::waitForClientConnection ( )

Ожидает подключения клиента.

Исключения

Error	Если возникает ошибка при принятии подключения клиента.

Объявления и описания членов классов находятся в файлах:

- code/server.h
- code/server.cpp

Глава 5

Файлы

5.1 cli.h

```
1 #pragma once
2
3 #include <string>
4 #include "error.h"
5 #include <iostream>
6 #include <cstring>
7
8 using namespace std;
9
16 class CLI
17 {
18 public:
24     CLI();
25
33     void parseArgs(int argc, char *argv[]);
34
38     void showHelp() const;
39
45     string getAddr() const;
46
52     int getPort() const;
53
59     string getLogFile() const;
60
66     string getDataFile() const;
67
68 private:
69     string addr;
70     int port;
71     string logf;
72     string data;
73 };
```

5.2 dataworker.h

```
1 #pragma once
2
3 #include <string>
4 #include <fstream>
5 #include <vector>
6 #include "error.h"
7
8 using namespace std;
9
16 class DataWorker
17 {
18 public:
24     explicit DataWorker(const string &filePath);
25
32     vector<pair<string, string>> readCredentials() const;
33
39     string getFilePath() const;
40
41 private:
42     string filePath;
43 };
```

5.3 error.h

```
1 #pragma once
2
3 #include <stdexcept>
4 #include <string>
5 #include <ctime>
6
7 using namespace std;
8
15 class Error : public runtime_error
16 {
17 public:
24     explicit Error(const string &what_arg, bool is_critical = false);
25
32     explicit Error(const char *what_arg, bool is_critical = false);
33
40     bool isCritical() const;
41
47     string getTimestamp() const;
48
49 private:
50     bool critical;
51     string timestamp;
52 };
```

5.4 logger.h

```
1 #pragma once
2
3 #include "error.h"
4 #include <string>
5 #include <fstream>
6
7 using namespace std;
8
9
16 class Logger
17 {
18 public:
25     explicit Logger(const string &logFilePath);
26
33     void write(const Error &exception);
34
40     string getLogFilePath() const;
41
42 private:
43     string logFilePath;
44     ofstream logFile;
45 };
```

5.5 md5.h

```
1 #define CRYPTOPP_ENABLE_NAMESPACE_WEAK 1
2 #pragma once
3
4 #include <string>
5 #include <sstream>
6 #include <cryptopp/hex.h>
7 #include <cryptopp/md5.h>
8 #include <cryptopp/osrng.h>
9 #include <cryptopp/filters.h>
10
11 using namespace std;
12 using namespace CryptoPP;
13 using namespace CryptoPP::Weak1;
20 std::string MD5(const std::string &data);
```

5.6 server.h

```
1 #pragma once
2
3 #include <string>
4 #include <vector>
5 #include <iostream>
6 #include <cstring>
```

```
7 #include <arpa/inet.h>
8 #include <unistd.h>
9 #include "error.h"
10 #include "md5.h"
11
12 using namespace std;
13
21 class Server
22 {
23 public:
31     Server(const std::string &addr, int port, const std::vector<std::pair<std::string, std::string» &data);
32
39     void initializeServer();
40
47     void terminateServer();
48
54     std::string &getAddr();
55
61     int &getPort();
62
68     std::vector<std::pair<std::string, std::string» &getData();
69
75     void waitForClientConnection();
76
83     void authenticateClient();
84
90     void calculateAndSendProduct();
91
92 private:
93     std::string addr;
94     int port;
95     std::vector<std::pair<std::string, std::string» data;
96     int socket;
97     int client;
98 };
```

Предметный указатель