

VServer

Создано системой Doxygen 1.9.4



1 Иерархический список классов	1
1.1 Иерархия классов	1
2 Алфавитный указатель классов	3
2.1 Классы	3
3 Список файлов	5
3.1 Файлы	5
4 Классы	7
4.1 Класс <code>AcceptConnectionError</code>	7
4.2 Класс <code>BindSocketError</code>	7
4.2.1 Подробное описание	7
4.2.2 Конструктор(ы)	7
4.2.2.1 <code>BindSocketError()</code> [1/2]	7
4.2.2.2 <code>BindSocketError()</code> [2/2]	8
4.3 Класс <code>CreateSocketError</code>	8
4.3.1 Подробное описание	9
4.3.2 Конструктор(ы)	9
4.3.2.1 <code>CreateSocketError()</code> [1/2]	9
4.3.2.2 <code>CreateSocketError()</code> [2/2]	9
4.4 Класс <code>InvalidArgumentError</code>	10
4.4.1 Подробное описание	10
4.4.2 Конструктор(ы)	10
4.4.2.1 <code>InvalidArgumentError()</code> [1/2]	10
4.4.2.2 <code>InvalidArgumentError()</code> [2/2]	10
4.5 Класс <code>InvalidAuthDataFormatError</code>	11
4.5.1 Подробное описание	11
4.5.2 Конструктор(ы)	11
4.5.2.1 <code>InvalidAuthDataFormatError()</code> [1/2]	11
4.5.2.2 <code>InvalidAuthDataFormatError()</code> [2/2]	12
4.6 Класс <code>InvalidLineFormat</code>	12
4.6.1 Подробное описание	13
4.6.2 Конструктор(ы)	13
4.6.2.1 <code>InvalidLineFormat()</code> [1/2]	13
4.6.2.2 <code>InvalidLineFormat()</code> [2/2]	13
4.7 Класс <code>InvalidLoginOrPasswordError</code>	14
4.7.1 Подробное описание	14
4.7.2 Конструктор(ы)	14
4.7.2.1 <code>InvalidLoginOrPasswordError()</code> [1/2]	14
4.7.2.2 <code>InvalidLoginOrPasswordError()</code> [2/2]	14
4.8 Класс <code>ListenSocketError</code>	15
4.8.1 Подробное описание	15
4.8.2 Конструктор(ы)	15

4.8.2.1 ListenSocketError() [1/2]	15
4.8.2.2 ListenSocketError() [2/2]	16
4.9 Класс Parser	16
4.9.1 Подробное описание	17
4.9.2 Методы	17
4.9.2.1 getAddress() [1/2]	17
4.9.2.2 getAddress() [2/2]	17
4.9.2.3 getPathToLog() [1/2]	18
4.9.2.4 getPathToLog() [2/2]	18
4.9.2.5 getPathToUsers() [1/2]	18
4.9.2.6 getPathToUsers() [2/2]	18
4.9.2.7 getPort() [1/2]	19
4.9.2.8 getPort() [2/2]	19
4.9.2.9 parse() [1/2]	19
4.9.2.10 parse() [2/2]	19
4.10 Класс ReadWriteError	20
4.10.1 Подробное описание	20
4.10.2 Конструктор(ы)	20
4.10.2.1 ReadWriteError() [1/2]	20
4.10.2.2 ReadWriteError() [2/2]	21
4.11 Класс RecvDataError	21
4.11.1 Подробное описание	22
4.11.2 Конструктор(ы)	22
4.11.2.1 RecvDataError() [1/2]	22
4.11.2.2 RecvDataError() [2/2]	22
4.12 Класс SendDataError	23
4.12.1 Подробное описание	23
4.12.2 Конструктор(ы)	23
4.12.2.1 SendDataError() [1/2]	23
4.12.2.2 SendDataError() [2/2]	23
4.13 Класс Server	24
4.13.1 Подробное описание	25
4.13.2 Конструктор(ы)	25
4.13.2.1 Server() [1/2]	25
4.13.2.2 Server() [2/2]	25
4.13.3 Методы	26
4.13.3.1 authentication() [1/2]	26
4.13.3.2 authentication() [2/2]	26
4.13.3.3 averange() [1/2]	26
4.13.3.4 averange() [2/2]	27
4.13.3.5 connect() [1/2]	27
4.13.3.6 connect() [2/2]	27
4.13.3.7 getAddress() [1/2]	27

4.13.3.8 getAddress() [2/2]	28
4.13.3.9 getPort() [1/2]	28
4.13.3.10 getPort() [2/2]	28
4.13.3.11 getUsers() [1/2]	29
4.13.3.12 getUsers() [2/2]	29
4.13.3.13 shutdown() [1/2]	29
4.13.3.14 shutdown() [2/2]	29
4.13.3.15 startup() [1/2]	30
4.13.3.16 startup() [2/2]	30
4.14 Класс ShutdownSocketError	30
4.14.1 Подробное описание	31
4.14.2 Конструктор(ы)	31
4.14.2.1 ShutdownSocketError() [1/2]	31
4.14.2.2 ShutdownSocketError() [2/2]	31
4.15 Класс UnknownError	32
4.15.1 Подробное описание	32
4.15.2 Конструктор(ы)	33
4.15.2.1 UnknownError() [1/2]	33
4.15.2.2 UnknownError() [2/2]	33
4.15.3 Методы	33
4.15.3.1 getFunc() [1/2]	33
4.15.3.2 getFunc() [2/2]	34
4.15.3.3 getMessage() [1/2]	34
4.15.3.4 getMessage() [2/2]	34
4.15.3.5 getName() [1/2]	34
4.15.3.6 getName() [2/2]	35
4.15.3.7 getTime() [1/2]	35
4.15.3.8 getTime() [2/2]	35
4.15.3.9 isCritical() [1/2]	35
4.15.3.10 isCritical() [2/2]	36
4.15.3.11 what() [1/2]	36
4.15.3.12 what() [2/2]	36
5 Файлы	37
5.1 errors.h	37
5.2 errors.h	38
5.3 arguments.h	40
5.4 arguments.h	40
5.5 rw.h	41
5.6 rw.h	41
5.7 sha1.h	41
5.8 sha1.h	41
5.9 server.h	42

5.10 server.h . . . . .	42
Предметный указатель	45

# Глава 1

## Иерархический список классов

### 1.1 Иерархия классов

Иерархия классов.

exception	
UnknownError . . . . .	32
AcceptConnectionError . . . . .	7
AcceptConnectionError . . . . .	7
BindSocketError . . . . .	7
BindSocketError . . . . .	7
CreateSocketError . . . . .	8
CreateSocketError . . . . .	8
InvalidArgumentError . . . . .	10
InvalidArgumentError . . . . .	10
InvalidAuthDataFormatError . . . . .	11
InvalidAuthDataFormatError . . . . .	11
InvalidLineFormat . . . . .	12
InvalidLineFormat . . . . .	12
InvalidLoginOrPasswordError . . . . .	14
InvalidLoginOrPasswordError . . . . .	14
ListenSocketError . . . . .	15
ListenSocketError . . . . .	15
ReadWriteError . . . . .	20
ReadWriteError . . . . .	20
RecvDataError . . . . .	21
RecvDataError . . . . .	21
SendDataError . . . . .	23
SendDataError . . . . .	23
ShutdownSocketError . . . . .	30
ShutdownSocketError . . . . .	30
UnknownError . . . . .	32
Parser . . . . .	16
Server . . . . .	24





## Глава 2

# Алфавитный указатель классов

### 2.1 Классы

Классы с их кратким описанием.

<a href="#">AcceptConnectionError</a>	
Ошибка принятия соединения . . . . .	7
<a href="#">BindSocketError</a>	
Ошибка привязки сокета . . . . .	7
<a href="#">CreateSocketError</a>	
Ошибка создания сокета . . . . .	8
<a href="#">InvalidArgumentError</a>	
Ошибка некорректного аргумента . . . . .	10
<a href="#">InvalidAuthDataFormatError</a>	
Ошибка некорректного формата данных аутентификации . . . . .	11
<a href="#">InvalidLineFormat</a>	
Ошибка некорректного формата строки . . . . .	12
<a href="#">InvalidLoginOrPasswordError</a>	
Ошибка некорректного логина или пароля . . . . .	14
<a href="#">ListenSocketError</a>	
Ошибка прослушивания сокета . . . . .	15
<a href="#">Parser</a>	
Класс для парсинга аргументов командной строки и хранения параметров . . . . .	16
<a href="#">ReadWriteError</a>	
Ошибка чтения/записи . . . . .	20
<a href="#">RecvDataError</a>	
Ошибка получения данных . . . . .	21
<a href="#">SendDataError</a>	
Ошибка отправки данных . . . . .	23
<a href="#">Server</a>	
Класс для представления сервера, способного обрабатывать подключения и аутентификацию пользователей . . . . .	24
<a href="#">ShutdownSocketError</a>	
Ошибка завершения работы сокета . . . . .	30
<a href="#">UnknownError</a>	
Базовый класс для всех ошибок . . . . .	32



## Глава 3

# Список файлов

### 3.1 Файлы

Полный список документированных файлов.

/home/student/Документы/busigin/MUBypass/doxygen/unit/arguments.h . . . . .	40
/home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h . . . . .	37
/home/student/Документы/busigin/MUBypass/doxygen/unit/rw.h . . . . .	41
/home/student/Документы/busigin/MUBypass/doxygen/unit/server.h . . . . .	42
/home/student/Документы/busigin/MUBypass/doxygen/unit/sha1.h . . . . .	41
/home/student/Документы/busigin/MUBypass/doxygen/code/arguments.h . . . . .	40
/home/student/Документы/busigin/MUBypass/doxygen/code/errors.h . . . . .	38
/home/student/Документы/busigin/MUBypass/doxygen/code/rw.h . . . . .	41
/home/student/Документы/busigin/MUBypass/doxygen/code/server.h . . . . .	42
/home/student/Документы/busigin/MUBypass/doxygen/code/sha1.h . . . . .	41



## Глава 4

# Классы

### 4.1 Класс AcceptConnectionError

Ошибка принятия соединения.

```
#include <errors.h>
```

Граф наследования: AcceptConnectionError:

### 4.2 Класс BindSocketError

Ошибка привязки сокета.

```
#include <errors.h>
```

Граф наследования: BindSocketError:

Граф связей класса BindSocketError:

Открытые члены

- [BindSocketError](#) (const string &func, bool is\_critical=false)  
Конструктор [BindSocketError](#).
- [BindSocketError](#) (const string &func, bool is\_critical=false)  
Конструктор [BindSocketError](#).

Дополнительные унаследованные члены

#### 4.2.1 Подробное описание

Ошибка привязки сокета.

#### 4.2.2 Конструктор(ы)

##### 4.2.2.1 BindSocketError() [1/2]

```
BindSocketError::BindSocketError (  
    const string & func,  
    bool is_critical = false )
```

Конструктор [BindSocketError](#).

## Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

## 4.2.2.2 BindSocketError() [2/2]

```
BindSocketError::BindSocketError (
    const string & func,
    bool is_critical = false )
```

Конструктор [BindSocketError](#).

## Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.cpp

## 4.3 Класс CreateSocketError

Ошибка создания сокета.

```
#include <errors.h>
```

Граф наследования: CreateSocketError:

Граф связей класса CreateSocketError:

## Открытые члены

- [CreateSocketError](#) (const string &func, bool is\_critical=false)  
Конструктор [CreateSocketError](#).
- [CreateSocketError](#) (const string &func, bool is\_critical=false)  
Конструктор [CreateSocketError](#).

## Дополнительные унаследованные члены

### 4.3.1 Подробное описание

Ошибка создания сокета.

### 4.3.2 Конструктор(ы)

#### 4.3.2.1 CreateSocketError() [1/2]

```
CreateSocketError::CreateSocketError (
    const string & func,
    bool is_critical = false )
```

Конструктор [CreateSocketError](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

#### 4.3.2.2 CreateSocketError() [2/2]

```
CreateSocketError::CreateSocketError (
    const string & func,
    bool is_critical = false )
```

Конструктор [CreateSocketError](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.cpp

## 4.4 Класс InvalidArgumentError

Ошибка некорректного аргумента.

```
#include <errors.h>
```

Граф наследования: InvalidArgumentError:

Граф связей класса InvalidArgumentError:

### Открытые члены

- [InvalidArgumentError](#) (const string &func, bool is\_critical=false)  
Конструктор [InvalidArgumentError](#).
- [InvalidArgumentError](#) (const string &func, bool is\_critical=false)  
Конструктор [InvalidArgumentError](#).

### Дополнительные унаследованные члены

#### 4.4.1 Подробное описание

Ошибка некорректного аргумента.

#### 4.4.2 Конструктор(ы)

##### 4.4.2.1 InvalidArgumentError() [1/2]

```
InvalidArgumentError::InvalidArgumentError (
    const string & func,
    bool is_critical = false )
```

Конструктор [InvalidArgumentError](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

##### 4.4.2.2 InvalidArgumentError() [2/2]

```
InvalidArgumentError::InvalidArgumentError (
    const string & func,
    bool is_critical = false )
```



Конструктор [InvalidArgumentError](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.cpp

## 4.5 Класс InvalidAuthDataFormatError

Ошибка некорректного формата данных аутентификации.

```
#include <errors.h>
```

Граф наследования:InvalidAuthDataFormatError:

Граф связей класса InvalidAuthDataFormatError:

Открытые члены

- [InvalidAuthDataFormatError](#) (const string &func, bool is\_critical=false)  
Конструктор [InvalidAuthDataFormatError](#).
- [InvalidAuthDataFormatError](#) (const string &func, bool is\_critical=false)  
Конструктор [InvalidAuthDataFormatError](#).

Дополнительные унаследованные члены

### 4.5.1 Подробное описание

Ошибка некорректного формата данных аутентификации.

### 4.5.2 Конструктор(ы)

#### 4.5.2.1 InvalidAuthDataFormatError() [1/2]

```
InvalidAuthDataFormatError::InvalidAuthDataFormatError (  
    const string & func,  
    bool is_critical = false )
```

Конструктор [InvalidAuthDataFormatError](#).

## Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

## 4.5.2.2 InvalidAuthDataFormatError() [2/2]

```
InvalidAuthDataFormatError::InvalidAuthDataFormatError (
    const string & func,
    bool is_critical = false )
```

Конструктор [InvalidAuthDataFormatError](#).

## Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.cpp

## 4.6 Класс InvalidLineFormat

Ошибка некорректного формата строки.

```
#include <errors.h>
```

Граф наследования:InvalidLineFormat:

Граф связей класса InvalidLineFormat:

## Открытые члены

- [InvalidLineFormat](#) (const string &func, bool is\_critical=false)  
Конструктор [InvalidLineFormat](#).
- [InvalidLineFormat](#) (const string &func, bool is\_critical=false)  
Конструктор [InvalidLineFormat](#).

## Дополнительные унаследованные члены

### 4.6.1 Подробное описание

Ошибка некорректного формата строки.

### 4.6.2 Конструктор(ы)

#### 4.6.2.1 InvalidLineFormat() [1/2]

```
InvalidLineFormat::InvalidLineFormat (
    const string & func,
    bool is_critical = false )
```

Конструктор [InvalidLineFormat](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

#### 4.6.2.2 InvalidLineFormat() [2/2]

```
InvalidLineFormat::InvalidLineFormat (
    const string & func,
    bool is_critical = false )
```

Конструктор [InvalidLineFormat](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.cpp

## 4.7 Класс InvalidLoginOrPasswordError

Ошибка некорректного логина или пароля.

```
#include <errors.h>
```

Граф наследования: InvalidLoginOrPasswordError:

Граф связей класса InvalidLoginOrPasswordError:

### Открытые члены

- [InvalidLoginOrPasswordError](#) (const string &func, bool is\_critical=false)  
Конструктор [InvalidLoginOrPasswordError](#).
- [InvalidLoginOrPasswordError](#) (const string &func, bool is\_critical=false)  
Конструктор [InvalidLoginOrPasswordError](#).

### Дополнительные унаследованные члены

#### 4.7.1 Подробное описание

Ошибка некорректного логина или пароля.

#### 4.7.2 Конструктор(ы)

##### 4.7.2.1 InvalidLoginOrPasswordError() [1/2]

```
InvalidLoginOrPasswordError::InvalidLoginOrPasswordError (
    const string & func,
    bool is_critical = false )
```

Конструктор [InvalidLoginOrPasswordError](#).

#### Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

##### 4.7.2.2 InvalidLoginOrPasswordError() [2/2]

```
InvalidLoginOrPasswordError::InvalidLoginOrPasswordError (
    const string & func,
    bool is_critical = false )
```

Конструктор [InvalidLoginOrPasswordError](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.cpp

## 4.8 Класс ListenSocketError

Ошибка прослушивания сокета.

```
#include <errors.h>
```

Граф наследования:ListenSocketError:

Граф связей класса ListenSocketError:

Открытые члены

- [ListenSocketError](#) (const string &func, bool is\_critical=false)  
Конструктор [ListenSocketError](#).
- [ListenSocketError](#) (const string &func, bool is\_critical=false)  
Конструктор [ListenSocketError](#).

Дополнительные унаследованные члены

### 4.8.1 Подробное описание

Ошибка прослушивания сокета.

### 4.8.2 Конструктор(ы)

#### 4.8.2.1 ListenSocketError() [1/2]

```
ListenSocketError::ListenSocketError (
    const string & func,
    bool is_critical = false )
```

Конструктор [ListenSocketError](#).

## Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

## 4.8.2.2 ListenSocketError() [2/2]

```
ListenSocketError::ListenSocketError (
    const string & func,
    bool is_critical = false )
```

Конструктор [ListenSocketError](#).

## Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.cpp

## 4.9 Класс Parser

Класс для парсинга аргументов командной строки и хранения параметров.

```
#include <arguments.h>
```

## Открытые члены

- Parser ()  
Конструктор по умолчанию. Устанавливает пути по умолчанию и параметры.
- void [parse](#) (int argc, char \*argv[])  
Парсит аргументы командной строки.
- void [help](#) () const  
Выводит сообщение помощи.
- std::string & [getPathToUsers](#) ()  
Возвращает путь к файлу с пользователями.
- std::string & [getPathToLog](#) ()  
Возвращает путь к файлу логов.
- std::string & [getAddress](#) ()  
Возвращает адрес.

- `int getPort () const`  
Возвращает порт.
- `Parser ()`  
Конструктор по умолчанию. Устанавливает пути по умолчанию и параметры.
- `void parse (int argc, char *argv[])`  
Парсит аргументы командной строки.
- `void help () const`  
Выводит сообщение помощи.
- `std::string & getPathToUsers ()`  
Возвращает путь к файлу с пользователями.
- `std::string & getPathToLog ()`  
Возвращает путь к файлу логов.
- `std::string & getAddress ()`  
Возвращает адрес.
- `int getPort () const`  
Возвращает порт.

#### 4.9.1 Подробное описание

Класс для парсинга аргументов командной строки и хранения параметров.

#### 4.9.2 Методы

##### 4.9.2.1 getAddress() [1/2]

`std::string & Parser::getAddress ( )`

Возвращает адрес.

Возвращает

`std::string&` Адрес.

##### 4.9.2.2 getAddress() [2/2]

`std::string & Parser::getAddress ( )`

Возвращает адрес.

Возвращает

`std::string&` Адрес.

#### 4.9.2.3 getPathToLog() [1/2]

`std::string & Parser::getPathToLog ( )`

Возвращает путь к файлу логов.

Возвращает

`std::string&` Путь к файлу логов.

#### 4.9.2.4 getPathToLog() [2/2]

`std::string & Parser::getPathToLog ( )`

Возвращает путь к файлу логов.

Возвращает

`std::string&` Путь к файлу логов.

#### 4.9.2.5 getPathToUsers() [1/2]

`std::string & Parser::getPathToUsers ( )`

Возвращает путь к файлу с пользователями.

Возвращает

`std::string&` Путь к файлу с пользователями.

#### 4.9.2.6 getPathToUsers() [2/2]

`std::string & Parser::getPathToUsers ( )`

Возвращает путь к файлу с пользователями.

Возвращает

`std::string&` Путь к файлу с пользователями.



4.9.2.7 `getPort()` [1/2]

```
int Parser::getPort ( ) const
```

Возвращает порт.

Возвращает

`int` Порт.

4.9.2.8 `getPort()` [2/2]

```
int Parser::getPort ( ) const
```

Возвращает порт.

Возвращает

`int` Порт.

4.9.2.9 `parse()` [1/2]

```
void Parser::parse (
    int argc,
    char * argv[] )
```

Парсит аргументы командной строки.

Аргументы

<code>argc</code>	Количество аргументов.
<code>argv</code>	Массив аргументов.

Исключения

<a href="#">InvalidArgumentError</a>	если указан неверный аргумент.
--------------------------------------	--------------------------------

4.9.2.10 `parse()` [2/2]

```
void Parser::parse (
    int argc,
    char * argv[] )
```

Парсит аргументы командной строки.

Аргументы

argc	Количество аргументов.
argv	Массив аргументов.

Исключения

<a href="#">InvalidArgumentError</a>	если указан неверный аргумент.
--------------------------------------	--------------------------------

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/arguments.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/arguments.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/arguments.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/arguments.cpp

## 4.10 Класс ReadWriteError

Ошибка чтения/записи.

```
#include <errors.h>
```

Граф наследования: ReadWriteError:

Граф связей класса ReadWriteError:

Открытые члены

- [ReadWriteError](#) (const string &func, bool is\_critical=false)  
Конструктор [ReadWriteError](#).
- [ReadWriteError](#) (const string &func, bool is\_critical=false)  
Конструктор [ReadWriteError](#).

Дополнительные унаследованные члены

### 4.10.1 Подробное описание

Ошибка чтения/записи.

### 4.10.2 Конструктор(ы)

#### 4.10.2.1 ReadWriteError() [1/2]

```
ReadWriteError::ReadWriteError (
    const string & func,
    bool is_critical = false )
```

Конструктор [ReadWriteError](#).

## Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

## 4.10.2.2 ReadWriteError() [2/2]

```
ReadWriteError::ReadWriteError (
    const string & func,
    bool is_critical = false )
```

Конструктор [ReadWriteError](#).

## Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.cpp

## 4.11 Класс RecvDataError

Ошибка получения данных.

```
#include <errors.h>
```

Граф наследования:RecvDataError:

Граф связей класса RecvDataError:

## Открытые члены

- [RecvDataError](#) (const string &func, bool is\_critical=false)  
Конструктор [RecvDataError](#).
- [RecvDataError](#) (const string &func, bool is\_critical=false)  
Конструктор [RecvDataError](#).

## Дополнительные унаследованные члены

### 4.11.1 Подробное описание

Ошибка получения данных.

### 4.11.2 Конструктор(ы)

#### 4.11.2.1 RecvDataError() [1/2]

```
RecvDataError::RecvDataError (
    const string & func,
    bool is_critical = false )
```

Конструктор [RecvDataError](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

#### 4.11.2.2 RecvDataError() [2/2]

```
RecvDataError::RecvDataError (
    const string & func,
    bool is_critical = false )
```

Конструктор [RecvDataError](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.cpp

## 4.12 Класс SendDataError

Ошибка отправки данных.

```
#include <errors.h>
```

Граф наследования:SendDataError:

Граф связей класса SendDataError:

### Открытые члены

- [SendDataError](#) (const string &func, bool is\_critical=false)  
Конструктор [SendDataError](#).
- [SendDataError](#) (const string &func, bool is\_critical=false)  
Конструктор [SendDataError](#).

### Дополнительные унаследованные члены

#### 4.12.1 Подробное описание

Ошибка отправки данных.

#### 4.12.2 Конструктор(ы)

##### 4.12.2.1 SendDataError() [1/2]

```
SendDataError::SendDataError (  
    const string & func,  
    bool is_critical = false )
```

Конструктор [SendDataError](#).

##### Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

##### 4.12.2.2 SendDataError() [2/2]

```
SendDataError::SendDataError (  
    const string & func,  
    bool is_critical = false )
```

Конструктор [SendDataError](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.cpp

## 4.13 Класс Server

Класс для представления сервера, способного обрабатывать подключения и аутентификацию пользователей.

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

Открытые члены

- [Server](#) (const string &address, uint16\_t port, const map< string, string > &users)  
Конструктор [Server](#).
- string & [getAddress](#) ()  
Возвращает адрес сервера.
- uint16\_t & [getPort](#) ()  
Возвращает порт сервера.
- map< string, string > & [getUsers](#) ()  
Возвращает логины и пароли пользователей.
- void [startup](#) ()  
Запускает сервер.
- void [shutdown](#) ()  
Останавливает сервер.
- void [connect](#) ()  
Ожидает подключения клиента.
- void [authentication](#) ()  
Аутентифицирует клиента.
- void [average](#) ()  
Вычисляет среднее арифметическое значений, полученных от клиента.
- [Server](#) (const string &address, uint16\_t port, const map< string, string > &users)  
Конструктор [Server](#).
- string & [getAddress](#) ()  
Возвращает адрес сервера.
- uint16\_t & [getPort](#) ()  
Возвращает порт сервера.
- map< string, string > & [getUsers](#) ()

- Возвращает логины и пароли пользователей.
- void `startup` ()  
Запускает сервер.
- void `shutdown` ()  
Останавливает сервер.
- void `connect` ()  
Ожидает подключения клиента.
- void `authentication` ()  
Аутентифицирует клиента.
- void `average` ()  
Вычисляет среднее арифметическое значений, полученных от клиента.

#### 4.13.1 Подробное описание

Класс для представления сервера, способного обрабатывать подключения и аутентификацию пользователей.

#### 4.13.2 Конструктор(ы)

##### 4.13.2.1 `Server()` [1/2]

```
Server::Server (
    const string & address,
    uint16_t port,
    const map< string, string > & users )
```

Конструктор `Server`.

Аргументы

address	Адрес сервера.
port	Порт сервера.
users	Логин и пароли пользователей.

##### 4.13.2.2 `Server()` [2/2]

```
Server::Server (
    const string & address,
    uint16_t port,
    const map< string, string > & users )
```

Конструктор `Server`.

## Аргументы

address	Адрес сервера.
port	Порт сервера.
users	Логины и пароли пользователей.

## 4.13.3 Методы

## 4.13.3.1 authentication() [1/2]

```
void Server::authentication ( )
```

Аутентифицирует клиента.

## Исключения

<a href="#">RecvDataError</a>	Если данные не могут быть получены от клиента.
<a href="#">InvalidAuthDataFormatError</a>	Если формат данных аутентификации некорректен.
<a href="#">InvalidLoginOrPasswordError</a>	Если логин или пароль некорректны.

## 4.13.3.2 authentication() [2/2]

```
void Server::authentication ( )
```

Аутентифицирует клиента.

## Исключения

<a href="#">RecvDataError</a>	Если данные не могут быть получены от клиента.
<a href="#">InvalidAuthDataFormatError</a>	Если формат данных аутентификации некорректен.
<a href="#">InvalidLoginOrPasswordError</a>	Если логин или пароль некорректны.

## 4.13.3.3 averange() [1/2]

```
void Server::averange ( )
```

Вычисляет среднее арифметическое значений, полученных от клиента.



## Исключения

<a href="#">RecvDataError</a>	Если данные не могут быть получены от клиента.
<a href="#">SendDataError</a>	Если данные не могут быть отправлены клиенту.

4.13.3.4 `average()` [2/2]

```
void Server::average ( )
```

Вычисляет среднее арифметическое значений, полученных от клиента.

## Исключения

<a href="#">RecvDataError</a>	Если данные не могут быть получены от клиента.
<a href="#">SendDataError</a>	Если данные не могут быть отправлены клиенту.

4.13.3.5 `connect()` [1/2]

```
void Server::connect ( )
```

Ожидает подключения клиента.

## Исключения

<a href="#">AcceptConnectionError</a>	Если подключение не может быть принято.
---------------------------------------	---

4.13.3.6 `connect()` [2/2]

```
void Server::connect ( )
```

Ожидает подключения клиента.

## Исключения

<a href="#">AcceptConnectionError</a>	Если подключение не может быть принято.
---------------------------------------	---

4.13.3.7 `getAddress()` [1/2]

```
string & Server::getAddress ( )
```

Возвращает адрес сервера.

Возвращает

`string&` Адрес сервера.

#### 4.13.3.8 `getAddress()` [2/2]

`string & Server::getAddress ( )`

Возвращает адрес сервера.

Возвращает

`string&` Адрес сервера.

#### 4.13.3.9 `getPort()` [1/2]

`uint16_t & Server::getPort ( )`

Возвращает порт сервера.

Возвращает

`uint16_t&` Порт сервера.

#### 4.13.3.10 `getPort()` [2/2]

`uint16_t & Server::getPort ( )`

Возвращает порт сервера.

Возвращает

`uint16_t&` Порт сервера.

## 4.13.3.11 getUsers() [1/2]

```
map< string, string > & Server::getUsers ( )
```

Возвращает логины и пароли пользователей.

Возвращает

```
map<string, string>& Логины и пароли пользователей.
```

## 4.13.3.12 getUsers() [2/2]

```
map< string, string > & Server::getUsers ( )
```

Возвращает логины и пароли пользователей.

Возвращает

```
map<string, string>& Логины и пароли пользователей.
```

## 4.13.3.13 shutdown() [1/2]

```
void Server::shutdown ( )
```

Останавливает сервер.

Исключения

<a href="#">ShutdownSocketError</a>	Если сокет не может быть корректно закрыт.
-------------------------------------	--

## 4.13.3.14 shutdown() [2/2]

```
void Server::shutdown ( )
```

Останавливает сервер.

Исключения

<a href="#">ShutdownSocketError</a>	Если сокет не может быть корректно закрыт.
-------------------------------------	--

4.13.3.15 `startup()` [1/2]

```
void Server::startup ( )
```

Запускает сервер.

Исключения

<a href="#">CreateSocketError</a>	Если сокет не может быть создан.
<a href="#">BindSocketError</a>	Если сокет не может быть привязан к адресу.
<a href="#">ListenSocketError</a>	Если сокет не может быть переведен в режим прослушивания.

4.13.3.16 `startup()` [2/2]

```
void Server::startup ( )
```

Запускает сервер.

Исключения

<a href="#">CreateSocketError</a>	Если сокет не может быть создан.
<a href="#">BindSocketError</a>	Если сокет не может быть привязан к адресу.
<a href="#">ListenSocketError</a>	Если сокет не может быть переведен в режим прослушивания.

Объявления и описания членов классов находятся в файлах:

- `/home/student/Документы/busigin/MUBypass/doxygen/code/server.h`
- `/home/student/Документы/busigin/MUBypass/doxygen/unit/server.h`
- `/home/student/Документы/busigin/MUBypass/doxygen/code/server.cpp`
- `/home/student/Документы/busigin/MUBypass/doxygen/unit/server.cpp`

4.14 Класс `ShutdownSocketError`

Ошибка завершения работы сокета.

```
#include <errors.h>
```

Граф наследования: `ShutdownSocketError`:

Граф связей класса `ShutdownSocketError`:

Открытые члены

- [ShutdownSocketError](#) (const string &func, bool is\_critical=false)  
Конструктор [ShutdownSocketError](#).
- [ShutdownSocketError](#) (const string &func, bool is\_critical=false)  
Конструктор [ShutdownSocketError](#).

## Дополнительные унаследованные члены

### 4.14.1 Подробное описание

Ошибка завершения работы сокета.

### 4.14.2 Конструктор(ы)

#### 4.14.2.1 ShutdownSocketError() [1/2]

```
ShutdownSocketError::ShutdownSocketError (
    const string & func,
    bool is_critical = false )
```

Конструктор [ShutdownSocketError](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

#### 4.14.2.2 ShutdownSocketError() [2/2]

```
ShutdownSocketError::ShutdownSocketError (
    const string & func,
    bool is_critical = false )
```

Конструктор [ShutdownSocketError](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.cpp

## 4.15 Класс UnknownError

Базовый класс для всех ошибок.

```
#include <errors.h>
```

Граф наследования: UnknownError:

Граф связей класса UnknownError:

### Открытые члены

- `UnknownError` (const string &func, bool is\_critical=false)  
Конструктор `UnknownError`.
- string `getName` () const  
Получает имя ошибки.
- string `getFunc` () const  
Получает имя функции, в которой возникла ошибка.
- string `getMessage` () const  
Получает сообщение об ошибке.
- time\_t `getTime` () const  
Получает время возникновения ошибки.
- bool `isCritical` () const  
Проверяет, является ли ошибка критической.
- const char \* `what` () const noexcept override  
Получает сообщение об ошибке для вывода.
- `UnknownError` (const string &func, bool is\_critical=false)  
Конструктор `UnknownError`.
- string `getName` () const  
Получает имя ошибки.
- string `getFunc` () const  
Получает имя функции, в которой возникла ошибка.
- string `getMessage` () const  
Получает сообщение об ошибке.
- time\_t `getTime` () const  
Получает время возникновения ошибки.
- bool `isCritical` () const  
Проверяет, является ли ошибка критической.
- const char \* `what` () const noexcept override  
Получает сообщение об ошибке для вывода.

### Защищенные данные

- string name = "UnknownError"  
Имя ошибки.

#### 4.15.1 Подробное описание

Базовый класс для всех ошибок.

### 4.15.2 Конструктор(ы)

#### 4.15.2.1 `UnknownError()` [1/2]

```
UnknownError::UnknownError (
    const string & func,
    bool is_critical = false )
```

Конструктор [UnknownError](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

#### 4.15.2.2 `UnknownError()` [2/2]

```
UnknownError::UnknownError (
    const string & func,
    bool is_critical = false )
```

Конструктор [UnknownError](#).

Аргументы

func	Имя функции, в которой возникла ошибка.
is_critical	Флаг критичности ошибки.

### 4.15.3 Методы

#### 4.15.3.1 `getFunc()` [1/2]

```
string UnknownError::getFunc ( ) const
```

Получает имя функции, в которой возникла ошибка.

Возвращает

Имя функции.

#### 4.15.3.2 getFunc() [2/2]

```
string UnknownError::getFunc ( ) const
```

Получает имя функции, в которой возникла ошибка.

Возвращает

Имя функции.

#### 4.15.3.3 getMessage() [1/2]

```
string UnknownError::getMessage ( ) const
```

Получает сообщение об ошибке.

Возвращает

Сообщение об ошибке.

#### 4.15.3.4 getMessage() [2/2]

```
string UnknownError::getMessage ( ) const
```

Получает сообщение об ошибке.

Возвращает

Сообщение об ошибке.

#### 4.15.3.5 getName() [1/2]

```
string UnknownError::getName ( ) const
```

Получает имя ошибки.

Возвращает

Имя ошибки.



## 4.15.3.6 getName() [2/2]

```
string UnknownError::getName ( ) const
```

Получает имя ошибки.

Возвращает

Имя ошибки.

## 4.15.3.7 getTime() [1/2]

```
time_t UnknownError::getTime ( ) const
```

Получает время возникновения ошибки.

Возвращает

Время возникновения ошибки.

## 4.15.3.8 getTime() [2/2]

```
time_t UnknownError::getTime ( ) const
```

Получает время возникновения ошибки.

Возвращает

Время возникновения ошибки.

## 4.15.3.9 isCritical() [1/2]

```
bool UnknownError::isCritical ( ) const
```

Проверяет, является ли ошибка критической.

Возвращает

true, если ошибка критическая, иначе false.

## 4.15.3.10 isCritical() [2/2]

```
bool UnknownError::isCritical ( ) const
```

Проверяет, является ли ошибка критической.

Возвращает

true, если ошибка критическая, иначе false.

## 4.15.3.11 what() [1/2]

```
const char * UnknownError::what ( ) const [override], [noexcept]
```

Получает сообщение об ошибке для вывода.

Возвращает

Сообщение об ошибке.

## 4.15.3.12 what() [2/2]

```
const char * UnknownError::what ( ) const [override], [noexcept]
```

Получает сообщение об ошибке для вывода.

Возвращает

Сообщение об ошибке.

Объявления и описания членов классов находятся в файлах:

- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.h
- /home/student/Документы/busigin/MUBypass/doxygen/code/errors.cpp
- /home/student/Документы/busigin/MUBypass/doxygen/unit/errors.cpp

## Глава 5

# Файлы

### 5.1 errors.h

```
1 #ifndef ERRORS_H
2 #define ERRORS_H
3
4 #include <string>
5 #include <ctime>
6 #include <exception>
7
8 using namespace std;
9
14 class UnknownError : public exception
15 {
16 public:
22     UnknownError(const string &func, bool is_critical = false);
23
28     string getName() const;
29
34     string getFunc() const;
35
40     string getMessage() const;
41
46     time_t getTime() const;
47
52     bool isCritical() const;
53
58     const char *what() const noexcept override;
59
60 protected:
61     string name = "UnknownError";
62
63 private:
64     string func;
65     time_t time;
66     bool is_critical;
67     mutable string message;
68 };
69
74 class InvalidArgumentError : public UnknownError
75 {
76 public:
82     InvalidArgumentError(const string &func, bool is_critical = false);
83 };
84
85 // Ошибки чтения/записи
86
91 class ReadWriteError : public UnknownError
92 {
93 public:
99     ReadWriteError(const string &func, bool is_critical = false);
100 };
101
106 class InvalidLineFormat : public UnknownError
107 {
108 public:
114     InvalidLineFormat(const string &func, bool is_critical = false);
115 };
116
117 // Ошибки сокета
118
```

```

123 class CreateSocketError : public UnknownError
124 {
125 public:
131     CreateSocketError(const string &func, bool is_critical = false);
132 };
133
138 class BindSocketError : public UnknownError
139 {
140 public:
146     BindSocketError(const string &func, bool is_critical = false);
147 };
148
153 class ListenSocketError : public UnknownError
154 {
155 public:
161     ListenSocketError(const string &func, bool is_critical = false);
162 };
163
168 class ShutdownSocketError : public UnknownError
169 {
170 public:
176     ShutdownSocketError(const string &func, bool is_critical = false);
177 };
178
183 class AcceptConnectionError : public UnknownError
184 {
185 public:
191     AcceptConnectionError(const string &func, bool is_critical = false);
192 };
193
198 class RecvDataError : public UnknownError
199 {
200 public:
206     RecvDataError(const string &func, bool is_critical = false);
207 };
208
213 class SendDataError : public UnknownError
214 {
215 public:
221     SendDataError(const string &func, bool is_critical = false);
222 };
223
224 // Ошибки аутентификации
225
230 class InvalidAuthDataFormatError : public UnknownError
231 {
232 public:
238     InvalidAuthDataFormatError(const string &func, bool is_critical = false);
239 };
240
245 class InvalidLoginOrPasswordError : public UnknownError
246 {
247 public:
253     InvalidLoginOrPasswordError(const string &func, bool is_critical = false);
254 };
255
256 #endif // ERRORS_H

```

## 5.2 errors.h

```

1 #ifndef ERRORS_H
2 #define ERRORS_H
3
4 #include <string>
5 #include <ctime>
6 #include <exception>
7
8 using namespace std;
9
14 class UnknownError : public exception
15 {
16 public:
22     UnknownError(const string &func, bool is_critical = false);
23
28     string getName() const;
29
34     string getFunc() const;
35
40     string getMessage() const;
41
46     time_t getTime() const;
47
52     bool isCritical() const;
53

```

```

58     const char *what() const noexcept override;
59
60 protected:
61     string name = "UnknownError";
62
63 private:
64     string func;
65     time_t time;
66     bool is_critical;
67     mutable string message;
68 };
69
70 class InvalidArgumentError : public UnknownError
71 {
72 public:
73     InvalidArgumentError(const string &func, bool is_critical = false);
74 };
75
76 // Ошибки чтения/записи
77
78 class ReadWriteError : public UnknownError
79 {
80 public:
81     ReadWriteError(const string &func, bool is_critical = false);
82 };
83
84 // Ошибки сокета
85
86 class CreateSocketError : public UnknownError
87 {
88 public:
89     CreateSocketError(const string &func, bool is_critical = false);
90 };
91
92 class BindSocketError : public UnknownError
93 {
94 public:
95     BindSocketError(const string &func, bool is_critical = false);
96 };
97
98 class ListenSocketError : public UnknownError
99 {
100 public:
101     ListenSocketError(const string &func, bool is_critical = false);
102 };
103
104 class ShutdownSocketError : public UnknownError
105 {
106 public:
107     ShutdownSocketError(const string &func, bool is_critical = false);
108 };
109
110 class AcceptConnectionError : public UnknownError
111 {
112 public:
113     AcceptConnectionError(const string &func, bool is_critical = false);
114 };
115
116 class RecvDataError : public UnknownError
117 {
118 public:
119     RecvDataError(const string &func, bool is_critical = false);
120 };
121
122 class SendDataError : public UnknownError
123 {
124 public:
125     SendDataError(const string &func, bool is_critical = false);
126 };
127
128 // Ошибки аутентификации
129
130 class InvalidAuthDataFormatError : public UnknownError
131 {
132 public:
133     InvalidAuthDataFormatError(const string &func, bool is_critical = false);
134 };
135
136 class InvalidLoginOrPasswordError : public UnknownError
137 {
138 public:

```

```

253     InvalidLoginOrPasswordError(const string &func, bool is_critical = false);
254 };
255
256 #endif // ERRORS_H

```

### 5.3 arguments.h

```

1 #ifndef PARSER_H
2 #define PARSER_H
3
4 #include <string>
5 #include <cstring>
6 #include <iostream>
7
8 #include "errors.h"
9
10 using namespace std;
11
12 class Parser
13 {
14 public:
15     Parser();
16
17     void parse(int argc, char *argv[]);
18
19     void help() const;
20
21     std::string& getPathToUsers();
22
23     std::string& getPathToLog();
24
25     std::string& getAddress();
26
27     int getPort() const;
28
29 private:
30     std::string pathToUsers;
31     std::string pathToLog;
32     std::string address;
33     int port;
34 };
35 #endif // PARSER_H

```

### 5.4 arguments.h

```

1 #ifndef PARSER_H
2 #define PARSER_H
3
4 #include <string>
5 #include <cstring>
6 #include <iostream>
7
8 #include "errors.h"
9
10 using namespace std;
11
12 class Parser
13 {
14 public:
15     Parser();
16
17     void parse(int argc, char *argv[]);
18
19     void help() const;
20
21     std::string& getPathToUsers();
22
23     std::string& getPathToLog();
24
25     std::string& getAddress();
26
27     int getPort() const;
28
29 private:
30     std::string pathToUsers;
31     std::string pathToLog;
32     std::string address;
33     int port;
34 };

```

```
72
73 #endif // PARSER_H
```

## 5.5 rw.h

```
1 #ifndef RW_H
2 #define RW_H
3
4 #include <map>
5 #include <string>
6
7 #include "errors.h"
8
9 using namespace std;
10
11 map<string, string> readUsers(const string &path);
12
13 void writeLog(const string &path, const UnknownError &e);
14
15 #endif // RW_H
```

## 5.6 rw.h

```
1 #ifndef RW_H
2 #define RW_H
3
4 #include <map>
5 #include <string>
6
7 #include "errors.h"
8
9 using namespace std;
10
11 map<string, string> readUsers(const string &path);
12
13 void writeLog(const string &path, const UnknownError &e);
14
15 #endif // RW_H
```

## 5.7 sha1.h

```
1 #ifndef SHA1_H
2 #define SHA1_H
3
4 #include <string>
5 #include <cryptopp/sha.h>
6 #include <cryptopp/hex.h>
7 #include <cryptopp/osrng.h>
8 #include <cryptopp/filters.h>
9
10 using namespace std;
11
12 string SHA1(const string &data);
13
14 #endif // SHA1_H
```

## 5.8 sha1.h

```
1 #ifndef SHA1_H
2 #define SHA1_H
3
4 #include <string>
5 #include <cryptopp/sha.h>
6 #include <cryptopp/hex.h>
7 #include <cryptopp/osrng.h>
8 #include <cryptopp/filters.h>
9
10 using namespace std;
11
12 string SHA1(const string &data);
13
14 #endif // SHA1_H
```

## 5.9 server.h

```

1 #ifndef SERVER_H
2 #define SERVER_H
3
4 #include "sha1.h"
5 #include "errors.h"
6
7 #include <map>
8 #include <string>
9 #include <vector>
10 #include <cstring>
11 #include <stdint>
12 #include <iostream>
13
14 #include <unistd.h>
15 #include <arpa/inet.h>
16 #include <sys/types.h>
17 #include <sys/socket.h>
18 #include <netinet/in.h>
19
20 using namespace std;
21
22 class Server
23 {
24 public:
25     Server(const string &address, uint16_t port, const map<string, string> &users);
26
27     string& getAddress();
28
29     uint16_t& getPort();
30
31     map<string, string>& getUsers();
32
33     void startup();
34
35     void shutdown();
36
37     void connect();
38
39     void authentication();
40
41     void averange();
42 private:
43     string address;
44     uint16_t port;
45     map<string, string> users;
46     int socket;
47     int client;
48 };
49 #endif // SERVER_H

```

## 5.10 server.h

```

1 #ifndef SERVER_H
2 #define SERVER_H
3
4 #include "sha1.h"
5 #include "errors.h"
6
7 #include <map>
8 #include <string>
9 #include <vector>
10 #include <cstring>
11 #include <stdint>
12 #include <iostream>
13
14 #include <unistd.h>
15 #include <arpa/inet.h>
16 #include <sys/types.h>
17 #include <sys/socket.h>
18 #include <netinet/in.h>
19
20 using namespace std;
21
22 class Server
23 {
24 public:
25     Server(const string &address, uint16_t port, const map<string, string> &users);
26
27     string& getAddress();

```



```
42
47  uint16_t& getPort();
48
53  map<string, string>& getUsers();
54
61  void startup();
62
67  void shutdown();
68
73  void connect();
74
81  void authentication();
82
88  void averange();
89
90 private:
91  string address;
92  uint16_t port;
93  map<string, string> users;
94  int socket;
95  int client;
96 };
97
98 #endif // SERVER_H
```



# Предметный указатель

/home/student/Документы/busigin/MUBypass/doxygen/Server/arguments.h, 40  
get Time 35  
/home/student/Документы/busigin/MUBypass/doxygen/Unit/server.h, 38  
get Users 29  
/home/student/Документы/busigin/MUBypass/doxygen/Server/rw.h, 41  
InvalidArgumentError, 10  
/home/student/Документы/busigin/MUBypass/doxygen/code/server.h, 42  
InvalidArgumentError, 10  
InvalidAuthDataFormatError, 11  
/home/student/Документы/busigin/MUBypass/doxygen/code/data.h, 41  
InvalidAuthDataFormatError, 11, 12  
InvalidLineFormat, 12  
/home/student/Документы/busigin/MUBypass/doxygen/unit/arguments.h, 40  
InvalidLineFormat, 13  
InvalidLoginOrPasswordError, 14  
/home/student/Документы/busigin/MUBypass/doxygen/unit/server.h, 37  
InvalidLoginOrPasswordError, 14  
isCritical 41  
/home/student/Документы/busigin/MUBypass/doxygen/unit/rw.h, 41  
UnknownError, 35  
/home/student/Документы/busigin/MUBypass/doxygen/unit/server.h, 42  
ListenSocketError, 15  
/home/student/Документы/busigin/MUBypass/doxygen/unit/shat.h, 41  
ListenSocketError, 15, 16  
parse  
Parser, 19  
Parser, 16  
getAddress, 17  
getPathToLog, 17, 18  
getPathToUsers, 18  
getPort, 18, 19  
parse, 19  
ReadWriteError, 20  
ReadWriteError, 20, 21  
RecvDataError, 21  
RecvDataError, 22  
SendDataError, 23  
SendDataError, 23  
Server, 24  
authentication, 26  
average, 26, 27  
connect, 27  
getAddress, 27, 28  
getPort, 28  
getUsers, 28, 29  
Server, 25  
shutdown, 29  
startup, 29, 30  
shutdown  
Server, 29  
ShutdownSocketError, 30  
ShutdownSocketError, 31  
AcceptConnectionError, 7  
authentication  
Server, 26  
average  
Server, 26, 27  
BindSocketError, 7  
BindSocketError, 7, 8  
connect  
Server, 27  
CreateSocketError, 8  
CreateSocketError, 9  
getAddress  
Parser, 17  
Server, 27, 28  
getFunc  
UnknownError, 33  
getMessage  
UnknownError, 34  
getName  
UnknownError, 34  
getPathToLog  
Parser, 17, 18  
getPathToUsers  
Parser, 18  
getPort  
Parser, 18, 19

startup

Server, [29](#), [30](#)

UnknownError, [32](#)

getFunc, [33](#)

getMessage, [34](#)

getName, [34](#)

getTime, [35](#)

isCritical, [35](#)

UnknownError, [33](#)

what, [36](#)

what

UnknownError, [36](#)