Meteologica server

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

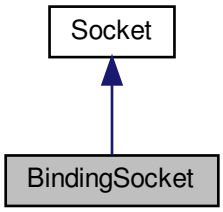
Chapter 3

Class Documentation

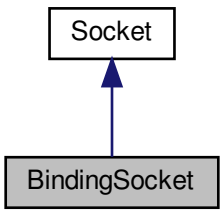
3.1 BindingSocket Class Reference

`#include <BindingSocket.hpp>`

Inheritance diagram for BindingSocket:



Collaboration diagram for BindingSocket:



Public Member Functions

- BindingSocket (int domain, int service, int protocol, int port, u_long interface, int bklg)

  Binding (server side) socket constructor. For more information, check https://man7.org/linux/man-pages/man2/socket.2.html.
- int connect_to_network (int sock, struct sockaddr_in address)

  Implements the connection to the network using the bind function.
- void start_listening ()

  Function to be called to start listening to the previously specified port.
- void test_connection ()

  Test function for the connection.
- void test_listening ()

  Test function for the port listening.
- int get_connection ()

  Getter for the connection binding, 0 on success, -1 on failure.
- int get_listening ()

  Getter for the listening, 0 on success, -1 on failure.

3.1.1 Detailed Description

Binding socket class. It inherits from the generic socket and implements the binding process, used in the server side.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BindingSocket()

```
BindingSocket::BindingSocket (
            int domain,
            int service,
            int protocol,
            int port,
            u_long interface,
            int bklg )
```

Binding (server side) socket constructor. For more information, check https://man7.org/linux/man-pages/man2/socket.2.html.

Parameters

domain	specifies the communicantion domain
service	specifies the communication semantics
protocol	specifies a particular protocol to be used with the socket
port	specifies the number of port that will be used
interface	specifies port interface
bklg	defines the maximum length to which the queue of pending connections may grow

3.1.3 Member Function Documentation

3.1.3.1 connect_to_network()

```
int BindingSocket::connect_to_network (
            int sock,
            struct sockaddr_in address )  [virtual]
```

Implements the connection to the network using the bind function.

Parameters

sock	specifies the socket file descriptor for the connection
service	address specifies the adress struct for the connection

Implements Socket.

The documentation for this class was generated from the following files:

- BindingSocket.hpp
- BindingSocket.cpp

3.2 Cache< Key, Value > Class Template Reference

```
#include <Cache.hpp>
```

Public Types

- using **value_type** = typename std::pair< Key, Value >
- using **value_it** = typename std::list< value_type >::iterator
- using **operation_guard** = typename std::lock_guard< std::mutex >

Public Member Functions

- Cache (size_t max_size)
  Cache constructor.
- void put (const Key &key, const Value &value)
  Puts a key and its value in the cache.
- Value & get (const Key &key)
  Queries a value from the cache.
- bool exists (const Key &key)
  Queries if a value is in the cache.
- size_t Size ()
  Queries the current size of the cache.
- void clear ()
  Clears the cache.

Friends

- std::ostream & operator<< (std::ostream &os, const Cache &c)

  Output operator. Writes the cache content.

3.2.1 Detailed Description

template<typename Key, typename Value>
class Cache< Key, Value >

Least recently used class. A template class to be used with any types. For more information check https://en.wikipedia.org/wiki/Cache_replacement_policies#Least_recently_used_(LRU)

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Cache()

```
template<typename Key, typename Value>
Cache< Key, Value >::Cache (
            size_t max_size )  [inline]
```

Cache constructor.

Parameters

max_size	specifies the maximum size of the cache. If 0, the cache size will be the maximum posible.

3.2.3 Member Function Documentation

3.2.3.1 exists()

```
template<typename Key, typename Value>
bool Cache< Key, Value >::exists (
            const Key & key )  [inline]
```

Queries if a value is in the cache.

Parameters

key	specifies the key to be queried

3.2.3.2 get()

```
template<typename Key, typename Value>
Value& Cache< Key, Value >::get (
            const Key & key )  [inline]
```

Queries a value from the cache.

Parameters

key	specifies the key to be queried

3.2.3.3 put()

```
template<typename Key, typename Value>
void Cache< Key, Value >::put (
            const Key & key,
            const Value & value )  [inline]
```

Puts a key and its value in the cache.

Parameters

key	specifies the key to be inserted
value	specifies the value to be inserted

The documentation for this class was generated from the following file:

- Cache.hpp

3.3 Server Class Reference

```
#include <Server.hpp>
```

Public Member Functions

- Server (int domain, int service, int protocol, int port, u_long interface, int bklg, std::size_t cache_size)

  Server constructor.
- ~Server ()

  Server destructor.

Friends

- class **ServerThreadPool**

3.3.1 Detailed Description

Server class that manages the connection to the socket and the tasks received

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Server()

```
Server::Server (
            int domain,
            int service,
            int protocol,
            int port,
            u_long interface,
            int bklg,
            std::size_t cache_size )
```

Server constructor.

Parameters

domain	specifies the communicantion domain for the socket
service	specifies the communication semantics for the socket
protocol	specifies a particular protocol to be used with the socket
port	specifies the number of port that will be used for the socket
interface	specifies port interface for the socket
bklg	defines the maximum length to which the queue of pending connections may grow in the socket
cache_size	defines the size of the cache

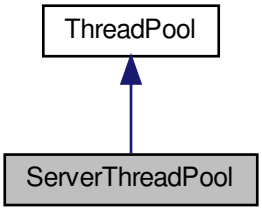
The documentation for this class was generated from the following files:

- Server.hpp
- Server.cpp

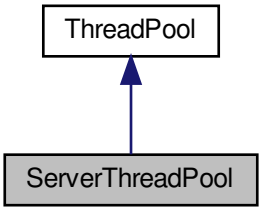
3.4 ServerThreadPool Class Reference

```
#include <ServerThreadPool.hpp>
```

Inheritance diagram for ServerThreadPool:



Collaboration diagram for ServerThreadPool:



Public Member Functions

- ServerThreadPool (std::size_t cache_size)
  Server thread pool constructor.
- ~ServerThreadPool ()
  Server thread pool constructor.
- void process_request (const std::pair< int, std::string > request)
  Process requests, returning the MD5 hash of the text and sleeping the specified time if the request is valid.
- void cache_clear ()
  Clears the cache.

3.4.1 Detailed Description

Thread pool class for the implemented server. It does implement the processing of petitions and includes the cache.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 ServerThreadPool()

```
ServerThreadPool::ServerThreadPool (
            std::size_t cache_size )
```

Server thread pool constructor.

Parameters

cache_size	specifies the maximum size of the cache. If 0, the cache size will be the maximum posible.

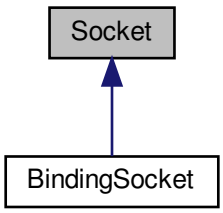
The documentation for this class was generated from the following files:

- ServerThreadPool.hpp
- ServerThreadPool.cpp

3.5 Socket Class Reference

```
#include <Socket.hpp>
```

Inheritance diagram for Socket:



Public Member Functions

- Socket (int domain, int service, int protocol, int port, u_long interface)

  Socket constructor. For more information, check https://man7.org/linux/man-pages/man2/socket.↩2.html.
- virtual int connect_to_network (int sock, struct sockaddr_in address)=0

  Virtual connection function. Must be implemented as bind (server) or connect (client).
- void test_sock ()

  Test function for the socket.
- int get_sock ()

  Getter function for the socket file descriptor.
- struct sockaddr_in get_address ()

  Getter function for the address struct.
- virtual ~Socket ()=default

  VIrtual default destructor.

3.5.1 Detailed Description

Generic socket class. It does not implement the connection process, as it is different in the server side and in the client side. In this project only the first one will be implemented, but a socket for the client side could also inherit from this class.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Socket()

```
Socket::Socket (
            int domain,
            int service,
            int protocol,
            int port,
            u_long interface )
```

Socket constructor. For more information, check https://man7.org/linux/man-pages/man2/socket.↩2.html.

Parameters

domain	specifies the communicantion domain
service	specifies the communication semantics
protocol	specifies a particular protocol to be used with the socket
port	specifies the number of port that will be used
interface	specifies port interface

3.5.3 Member Function Documentation

3.5.3.1 connect_to_network()

```
virtual int Socket::connect_to_network (
            int sock,
            struct sockaddr_in address )  [pure virtual]
```

Virtual connection function. Must be implemented as bind (server) or connect (client).

Parameters

sock	specifies the socket file descriptor for the connection
service	address specifies the adress struct for the connection

Implemented in BindingSocket.

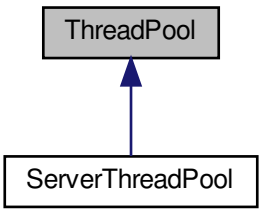
The documentation for this class was generated from the following files:

- Socket.hpp
- Socket.cpp

3.6 ThreadPool Class Reference

`#include <ThreadPool.hpp>`

Inheritance diagram for ThreadPool:



Public Member Functions

- ThreadPool ()
  Thread pool constructor.
- ~ThreadPool ()
  Thread pool destructor.
- void queue_work (int fd, std::string &request)
  Add work to the thread pool queue.

3.6.1 Detailed Description

Generic thread pool class. It does not implement the processing of the requests.

The documentation for this class was generated from the following files:

- ThreadPool.hpp
- ThreadPool.cpp

Index