r-type

0.0.0

Generated by Doxygen 1.11.0

Chapter 1

r-type

1.1 R-Type

The Goal of this project is to implement a multithreaded server and a graphical client for a game called R-Type, using an engine of your own design.

1.1.1 Supported Platforms

Platform	Compiler	Status
Linux	g++	
macOS	g++	
Windows	MSVC	

1.1.2 Project Structure

```
R-Type
    assets              # Game assets (images, sounds, etc.)
    client              # Client source code
    documentation          # Project documentation
    modules             # Static libraries for the project
    scripts             # Build and utility scripts
    server              # Server source code
    tests               # Unit and integration tests
    third-party            # External libraries as submodules
```

1.1.3 Prerequisites

Make sure you have the following dependencies installed on your system:

- CMake 4.0.0
- C++23

1.1.4 Clone the project

Important

When cloning the project, you should also initialize the submodules:
`git clone --recurse-submodules git@github.com:bobis33/R-Type.git`

If you already cloned the project, you can initialize the submodules with:
`git submodule update --init --recursive`

1.1.5 Build and Run

1.1.5.1 Unix (Linux, macOS)

```
./scripts/unix/build.sh release
## Or
cmake -S . -B cmake-build-release -G "Ninja" -DCMAKE_BUILD_TYPE=Release -DCMAKE_CXX_COMPILER=g++
      -DCMAKE_C_COMPILER=gcc
cmake --build cmake-build-release -- -j4
## Then
./cmake-build-release/r-type_client ## client
./cmake-build-release/r-type_server ## server
```

1.1.5.2 Windows

```
cmake -S . -B cmake-build-release -G "Visual Studio 17 2022" -A x64 -DCMAKE_BUILD_TYPE=Release
cmake --build cmake-build-release --config Release
## Then
cmake-build-release\bin\r-type_client.exe ## client
cmake-build-release\bin\r-type_server.exe ## server
```

For more details:

- Client documentation
- Server documentation

1.1.6 External Libraries

All dependencies are included as submodules in the third-party directory.

1.1.7 Commit Norms

Commit Type	Description
build	Changes that affect the build system or external dependencies (npm, make, etc.)
ci	Changes related to integration files and scripts or configuration (Travis, Ansible, BrowserStack, etc.)
feat	Addition of a new feature
fix	Bug fix
perf	Performance improvements
refactor	Modification that neither adds a new feature nor improves performance
style	Change that does not affect functionality or semantics (indentation, formatting, adding space, renaming a variable, etc.)
docs	Writing or updating documentation
test	Addition or modification of tests

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 rtp Namespace Reference

Classes

- struct ArgsConfig
- class ArgsHandler

  Class to handle command line arguments.
- class Client

  Class for the client.
- struct EnvConfig
- class EventHandler

  Class for the EventHandler.
- class Renderer

  Class for the renderer.
- class Server

  Class for the server.

Typedefs

- using json = nlohmann::json

5.1.1 Typedef Documentation

5.1.1.1 json

typedef nlohmann::json rtp::json = nlohmann::json

Definition at line 16 of file ArgsHandler.hpp.

5.2 sf Namespace Reference

5.3 utl Namespace Reference

Classes

- class Clock
  Class for clock.
- class Logger

Enumerations

- enum class LogLevel : uint8_t { INFO , WARNING }

Functions

- std::vector< char > readFile (const std::string &filename)

5.3.1 Enumeration Type Documentation

5.3.1.1 LogLevel

enum class utl::LogLevel : uint8_t [strong]

Enumerator

INFO	
WARNING	

Definition at line 11 of file Logger.hpp.

5.3.2 Function Documentation

5.3.2.1 readFile()

std::vector< char > utl::readFile (
const std::string & filename) [nodiscard]

Definition at line 5 of file utils.cpp.

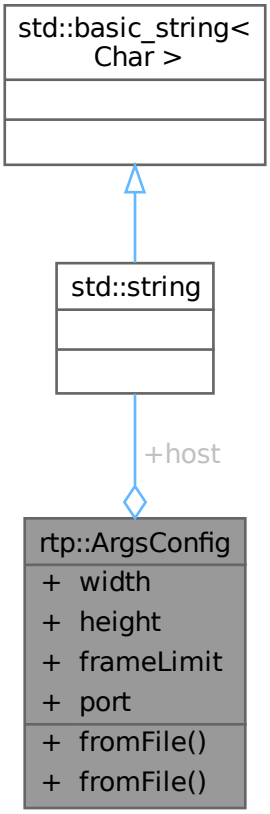
Chapter 6

Class Documentation

6.1 rtp::ArgsConfig Struct Reference

#include <ArgsHandler.hpp>

Collaboration diagram for rtp::ArgsConfig:



Static Public Member Functions

- static ArgsConfig fromFile (const std::string &path)
- static ArgsConfig fromFile (const std::string &path)

Public Attributes

- unsigned int width = 960
- unsigned int height = 540
- unsigned int frameLimit = 240
- std::string host = "0.0.0.0"
- unsigned int port = 2560

6.1.1 Detailed Description

Definition at line 18 of file ArgsHandler.hpp.

6.1.2 Member Function Documentation

6.1.2.1 fromFile() [1/2]

static ArgsConfig rtp::ArgsConfig::fromFile (
const std::string & path) [inline], [static]

Definition at line 24 of file ArgsHandler.hpp.

References frameLimit, height, and width.

Referenced by rtp::ArgsHandler::ParseArgs().

Here is the caller graph for this function:

6.1.2.2 fromFile() [2/2]

static ArgsConfig rtp::ArgsConfig::fromFile (
const std::string & path) [inline], [static]

Definition at line 23 of file ArgsHandler.hpp.

References host, and port.

6.1.3 Member Data Documentation

6.1.3.1 frameLimit

unsigned int rtp::ArgsConfig::frameLimit = 240

Definition at line 22 of file ArgsHandler.hpp.

Referenced by fromFile().

6.1.3.2 height

unsigned int rtp::ArgsConfig::height = 540

Definition at line 21 of file ArgsHandler.hpp.

Referenced by fromFile().

6.1.3.3 host

std::string rtp::ArgsConfig::host = "0.0.0.0"

Definition at line 20 of file ArgsHandler.hpp.

Referenced by fromFile().

6.1.3.4 port

unsigned int rtp::ArgsConfig::port = 2560

Definition at line 21 of file ArgsHandler.hpp.

Referenced by fromFile().

6.1.3.5 width

unsigned int rtp::ArgsConfig::width = 960

Definition at line 20 of file ArgsHandler.hpp.

Referenced by fromFile().

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

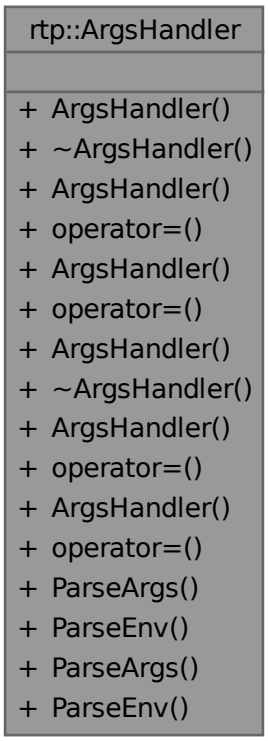
- /home/masina/Projects/Epitech/rtype/client/include/R-Type/ArgsHandler.hpp
- /home/masina/Projects/Epitech/rtype/server/include/R-Type/ArgsHandler.hpp

6.2 rtp::ArgsHandler Class Reference

Class to handle command line arguments.

#include <ArgsHandler.hpp>

Collaboration diagram for rtp::ArgsHandler:



Public Member Functions

- ArgsHandler ()=default
- ~ArgsHandler ()=default
- ArgsHandler (const ArgsHandler &)=delete
- ArgsHandler & operator= (const ArgsHandler &)=delete
- ArgsHandler (ArgsHandler &&)=delete
- ArgsHandler & operator= (ArgsHandler &&)=delete
- ArgsHandler ()=default
- ~ArgsHandler ()=default
- ArgsHandler (const ArgsHandler &)=delete
- ArgsHandler & operator= (const ArgsHandler &)=delete
- ArgsHandler (ArgsHandler &&)=delete
- ArgsHandler & operator= (ArgsHandler &&)=delete

Static Public Member Functions

- static ArgsConfig ParseArgs (int argc, const char *const argv[])
- static EnvConfig ParseEnv (const char *const env[])
- static ArgsConfig ParseArgs (int argc, const char *const argv[])
- static EnvConfig ParseEnv (const char *const env[])

6.2.1 Detailed Description

Class to handle command line arguments.

Definition at line 51 of file ArgsHandler.hpp.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 ArgsHandler() [1/6]

rtp::ArgsHandler::ArgsHandler () [default]

6.2.2.2 ~ArgsHandler() [1/2]

rtp::ArgsHandler::~ArgsHandler () [default]

6.2.2.3 ArgsHandler() [2/6]

rtp::ArgsHandler::ArgsHandler (
const ArgsHandler &) [delete]

6.2.2.4 ArgsHandler() [3/6]

rtp::ArgsHandler::ArgsHandler (
ArgsHandler &&) [delete]

6.2.2.5 ArgsHandler() [4/6]

rtp::ArgsHandler::ArgsHandler () [default]

6.2.2.6 ~ArgsHandler() [2/2]

rtp::ArgsHandler::~ArgsHandler () [default]

6.2.2.7 ArgsHandler() [5/6]

rtp::ArgsHandler::ArgsHandler (
const ArgsHandler &) [delete]

6.2.2.8 ArgsHandler() [6/6]

rtp::ArgsHandler::ArgsHandler (
ArgsHandler &&) [delete]

6.2.3 Member Function Documentation

6.2.3.1 operator=() [1/4]

ArgsHandler & rtp::ArgsHandler::operator= (
ArgsHandler &&) [delete]

6.2.3.2 operator=() [2/4]

ArgsHandler & rtp::ArgsHandler::operator= (
ArgsHandler &&) [delete]

6.2.3.3 operator=() [3/4]

ArgsHandler & rtp::ArgsHandler::operator= (
const ArgsHandler &) [delete]

6.2.3.4 operator=() [4/4]

ArgsHandler & rtp::ArgsHandler::operator= (
const ArgsHandler &) [delete]

6.2.3.5 ParseArgs() [1/2]

rtp::ArgsConfig rtp::ArgsHandler::ParseArgs (
int argc,
const char *const argv[]) [static]

Definition at line 25 of file argsHandler.cpp.

References rtp::ArgsConfig::fromFile(), HELP_MESSAGE, utl::INFO, utl::Logger::log(), and VERSION_MESSAGE.

Referenced by main().

Here is the call graph for this function:

Here is the caller graph for this function:

6.2.3.6 ParseArgs() [2/2]

static ArgsConfig rtp::ArgsHandler::ParseArgs (
int argc,
const char *const argv[]) [static]

6.2.3.7 ParseEnv() [1/2]

rtp::EnvConfig rtp::ArgsHandler::ParseEnv (
const char *const env[]) [static]

Definition at line 62 of file argsHandler.cpp.

Referenced by main().

Here is the caller graph for this function:

6.2.3.8 ParseEnv() [2/2]

static EnvConfig rtp::ArgsHandler::ParseEnv (
const char *const env[]) [static]

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

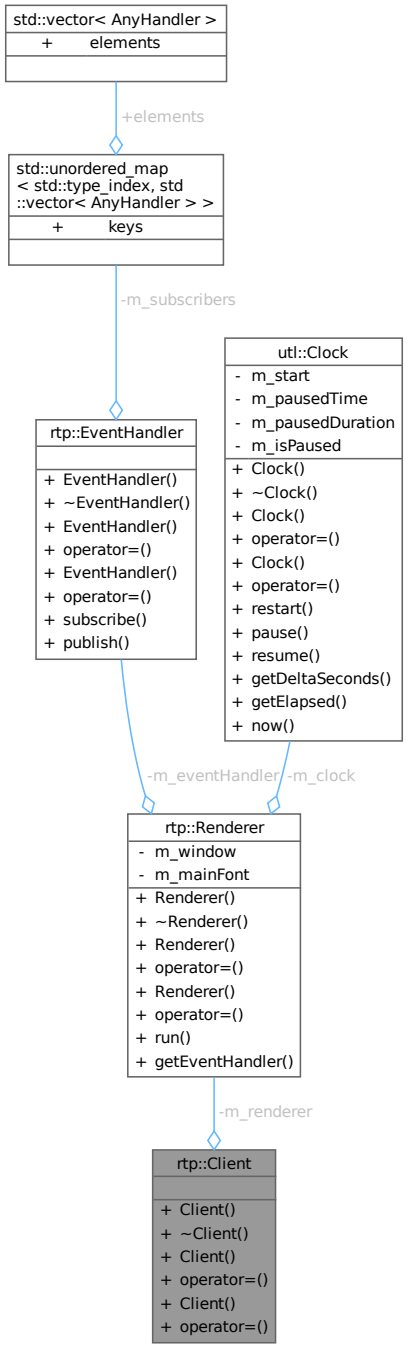
- /home/masina/Projects/Epitech/rtype/client/include/R-Type/ArgsHandler.hpp
- /home/masina/Projects/Epitech/rtype/server/include/R-Type/ArgsHandler.hpp
- /home/masina/Projects/Epitech/rtype/client/src/argsHandler.cpp
- /home/masina/Projects/Epitech/rtype/server/src/argsHandler.cpp

6.3 rtp::Client Class Reference

Class for the client.

#include <Client.hpp>

Collaboration diagram for rtp::Client:



Public Member Functions

- Client (ArgsConfig cfg)
- ~Client ()=default
- Client (const Client &)=delete
- Client & operator= (const Client &)=delete
- Client (Client &&)=delete
- Client & operator= (Client &&)=delete

Private Attributes

- Renderer m_renderer

6.3.1 Detailed Description

Class for the client.

Definition at line 20 of file Client.hpp.

6.3.2 Constructor & Destructor Documentation

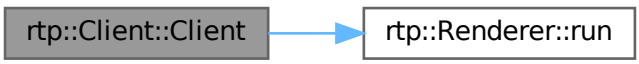
6.3.2.1 Client() [1/3]

rtp::Client::Client (
ArgsConfig cfg) [explicit]

Definition at line 3 of file client.cpp.

References m_renderer, and rtp::Renderer::run().

Here is the call graph for this function:



6.3.2.2 ~Client()

rtp::Client::~Client () [default]

6.3.2.3 Client() [2/3]

rtp::Client::Client (
const Client &) [delete]

6.3.2.4 Client() [3/3]

rtp::Client::Client (
Client &&) [delete]

6.3.3 Member Function Documentation

6.3.3.1 operator=() [1/2]

Client & rtp::Client::operator= (
Client &&) [delete]

6.3.3.2 operator=() [2/2]

Client & rtp::Client::operator= (
const Client &) [delete]

6.3.4 Member Data Documentation

6.3.4.1 m_renderer

Renderer rtp::Client::m_renderer [private]

Definition at line 33 of file Client.hpp.

Referenced by Client().

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

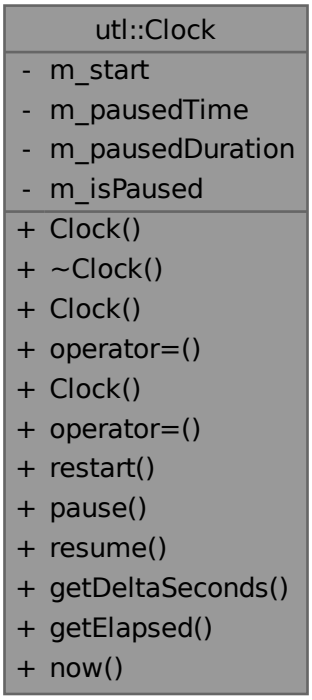
- /home/masina/Projects/Epitech/rtype/client/include/R-Type/Client.hpp
- /home/masina/Projects/Epitech/rtype/client/src/client.cpp

6.4 utl::Clock Class Reference

Class for clock.

#include <Clock.hpp>

Collaboration diagram for utl::Clock:



Public Types

- using TimePoint = std::chrono::time_point<std::chrono::high_resolution_clock>

Public Member Functions

- Clock (const bool startNow=true)
- ~Clock ()=default
- Clock (const Clock &)=delete
- Clock & operator= (const Clock &)=delete
- Clock (Clock &&)=delete
- Clock & operator= (Clock &&)=delete
- void restart ()
- void pause ()
- void resume ()
- float getDeltaSeconds () const
- template<typename Duration = std::chrono::seconds>
  auto getElapsed () const

Static Public Member Functions

- static TimePoint now ()

Private Types

- using Duration = std::chrono::high_resolution_clock::duration

Private Attributes

- TimePoint m_start
- TimePoint m_pausedTime
- Duration m_pausedDuration
- bool m_isPaused {false}

Friends

- std::ostream & operator<< (std::ostream &os, const Clock &clock)

6.4.1 Detailed Description

Class for clock.

Definition at line 20 of file Clock.hpp.

6.4.2 Member Typedef Documentation

6.4.2.1 Duration

using utl::Clock::Duration = std::chrono::high_resolution_clock::duration [private]

Definition at line 78 of file Clock.hpp.

6.4.2.2 TimePoint

using utl::Clock::TimePoint = std::chrono::time_point<std::chrono::high_resolution_clock>

Definition at line 24 of file Clock.hpp.

6.4.3 Constructor & Destructor Documentation

6.4.3.1 Clock() [1/3]

utl::Clock::Clock (
const bool startNow = true) [inline], [explicit]

Definition at line 26 of file Clock.hpp.

6.4.3.2 ~Clock()

utl::Clock::~Clock () [default]

6.4.3.3 Clock() [2/3]

utl::Clock::Clock (
const Clock &) [delete]

6.4.3.4 Clock() [3/3]

utl::Clock::Clock (
Clock &&) [delete]

6.4.4 Member Function Documentation

6.4.4.1 getDeltaSeconds()

float utl::Clock::getDeltaSeconds () const [inline], [nodiscard]

Definition at line 63 of file Clock.hpp.

References m_isPaused, m_pausedDuration, m_pausedTime, m_start, and now().

Here is the call graph for this function:

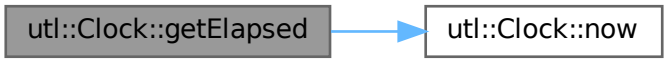
6.4.4.2 getElapsed()

template<typename Duration = std::chrono::seconds>
auto utl::Clock::getElapsed () const [inline], [nodiscard]

Definition at line 72 of file Clock.hpp.

References m_pausedDuration, m_start, and now().

Here is the call graph for this function:



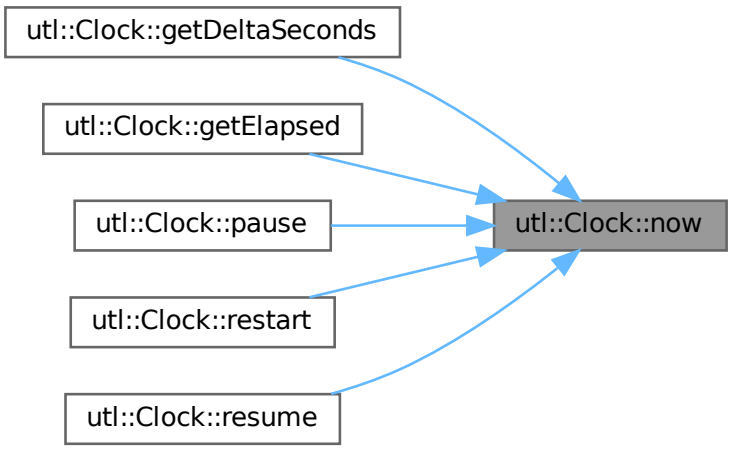
6.4.4.3 now()

static TimePoint utl::Clock::now () [inline], [static]

Definition at line 40 of file Clock.hpp.

Referenced by getDeltaSeconds(), getElapsed(), pause(), restart(), and resume().

Here is the caller graph for this function:



6.4.4.4 operator=() [1/2]

Clock & utl::Clock::operator= (
Clock &&) [delete]

6.4.4.5 operator=() [2/2]

Clock & utl::Clock::operator= (
const Clock &) [delete]

6.4.4.6 pause()

void utl::Clock::pause () [inline]

Definition at line 47 of file Clock.hpp.

References m_isPaused, m_pausedTime, and now().

Here is the call graph for this function:

6.4.4.7 restart()

void utl::Clock::restart () [inline]

Definition at line 41 of file Clock.hpp.

References m_isPaused, m_pausedDuration, m_start, and now().

Here is the call graph for this function:

6.4.4.8 resume()

void utl::Clock::resume () [inline]

Definition at line 55 of file Clock.hpp.

References m_isPaused, m_pausedDuration, m_pausedTime, and now().

Here is the call graph for this function:

6.4.5 Friends And Related Symbol Documentation

6.4.5.1 operator<<

std::ostream & operator<< (
std::ostream & os,
const Clock & clock) [friend]

Definition at line 34 of file Clock.hpp.

6.4.6 Member Data Documentation

6.4.6.1 m_isPaused

bool utl::Clock::m_isPaused {false} [private]

Definition at line 83 of file Clock.hpp.

Referenced by getDeltaSeconds(), pause(), restart(), and resume().

6.4.6.2 m_pausedDuration

Duration utl::Clock::m_pausedDuration [private]

Definition at line 82 of file Clock.hpp.

Referenced by getDeltaSeconds(), getElapsed(), restart(), and resume().

6.4.6.3 m_pausedTime

TimePoint utl::Clock::m_pausedTime [private]

Definition at line 81 of file Clock.hpp.

Referenced by getDeltaSeconds(), pause(), and resume().

6.4.6.4 m_start

TimePoint utl::Clock::m_start [private]

Definition at line 80 of file Clock.hpp.

Referenced by getDeltaSeconds(), getElapsed(), and restart().

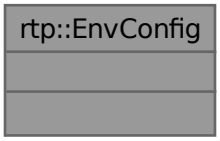
The documentation for this class was generated from the following file:

- /home/masina/Projects/Epitech/rtype/modules/Utils/include/Utils/Clock.hpp

6.5 rtp::EnvConfig Struct Reference

#include <ArgsHandler.hpp>

Collaboration diagram for rtp::EnvConfig:



6.5.1 Detailed Description

Definition at line 42 of file ArgsHandler.hpp.

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

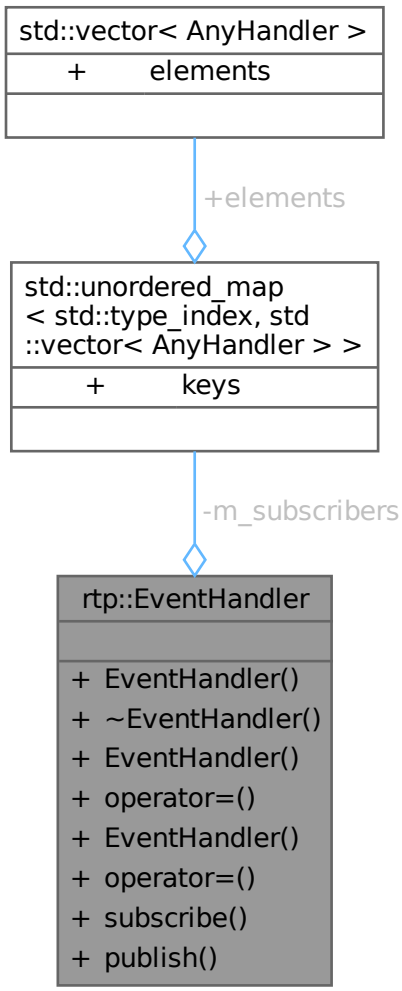
- /home/masina/Projects/Epitech/rtype/client/include/R-Type/ArgsHandler.hpp
- /home/masina/Projects/Epitech/rtype/server/include/R-Type/ArgsHandler.hpp

6.6 rtp::EventHandler Class Reference

Class for the EventHandler.

#include <EventHandler.hpp>

Collaboration diagram for rtp::EventHandler:



Public Types

- template<typename T >
  using Handler = std::function<void(const T &)>

Public Member Functions

- EventHandler ()=default
- ~EventHandler ()=default
- EventHandler (const EventHandler &)=delete
- EventHandler & operator= (const EventHandler &)=delete
- EventHandler (EventHandler &&)=delete
- EventHandler & operator= (EventHandler &&)=delete
- template<typename T >
  void subscribe (Handler< T > handler)
- void publish (const sf::Event &e)

Private Types

- using AnyHandler = std::function<void(const sf::Event &)>

Private Attributes

- std::unordered_map< std::type_index, std::vector< AnyHandler > > m_subscribers

6.6.1 Detailed Description

Class for the EventHandler.

Definition at line 21 of file EventHandler.hpp.

6.6.2 Member Typedef Documentation

6.6.2.1 AnyHandler

using rtp::EventHandler::AnyHandler = std::function<void(const sf::Event &)> [private]

Definition at line 51 of file EventHandler.hpp.

6.6.2.2 Handler

template<typename T >
using rtp::EventHandler::Handler = std::function<void(const T &)>

Definition at line 25 of file EventHandler.hpp.

6.6.3 Constructor & Destructor Documentation

6.6.3.1 EventHandler() [1/3]

rtp::EventHandler::EventHandler () [default]

6.6.3.2 ~EventHandler()

rtp::EventHandler::~EventHandler () [default]

6.6.3.3 EventHandler() [2/3]

rtp::EventHandler::EventHandler (
const EventHandler &) [delete]

6.6.3.4 EventHandler() [3/3]

rtp::EventHandler::EventHandler (
EventHandler &&) [delete]

6.6.4 Member Function Documentation

6.6.4.1 operator=() [1/2]

EventHandler & rtp::EventHandler::operator= (
const EventHandler &) [delete]

6.6.4.2 operator=() [2/2]

EventHandler & rtp::EventHandler::operator= (
EventHandler &&) [delete]

6.6.4.3 publish()

void rtp::EventHandler::publish (
const sf::Event & e)

Definition at line 6 of file eventHandler.cpp.

References utl::Logger::log(), m_subscribers, and utl::WARNING.

Here is the call graph for this function:

6.6.4.4 subscribe()

template<typename T >
void rtp::EventHandler::subscribe (
Handler< T > handler) [inline]

Definition at line 35 of file EventHandler.hpp.

References m_subscribers.

6.6.5 Member Data Documentation

6.6.5.1 m_subscribers

std::unordered_map<std::type_index, std::vector<AnyHandler> > rtp::EventHandler::m_subscribers [private]

Definition at line 52 of file EventHandler.hpp.

Referenced by publish(), and subscribe().

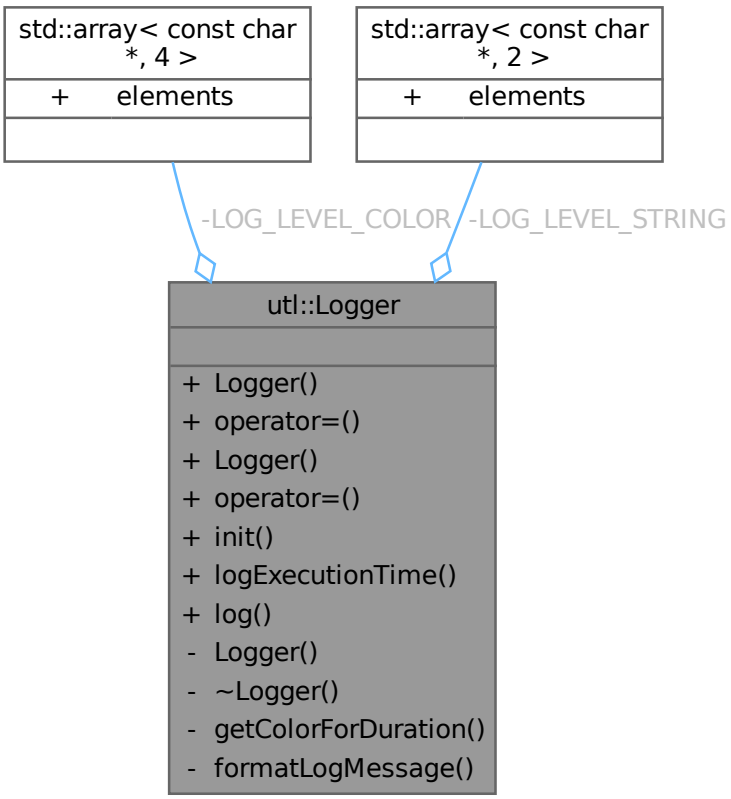
The documentation for this class was generated from the following files:

- /home/masina/Projects/Epitech/rtype/client/include/R-Type/Renderer/EventHandler.hpp
- /home/masina/Projects/Epitech/rtype/client/src/renderer/eventHandler.cpp

6.7 utl::Logger Class Reference

#include <Logger.hpp>

Collaboration diagram for utl::Logger:



Public Member Functions

- Logger (const Logger &)=delete
- Logger & operator= (const Logger &)=delete
- Logger (Logger &&)=delete
- Logger & operator= (Logger &&)=delete

Static Public Member Functions

- static void init ()
- template<typename Func >
  static void logExecutionTime (const std::string &message, Func &&func)
- static void log (const std::string &message, const LogLevel &logLevel)

Private Types

- enum ColorIndex : uint8_t { COLOR_ERROR , COLOR_INFO , COLOR_WARNING , COLOR_RESET }

Private Member Functions

- Logger ()=default
- ~Logger ()=default

Static Private Member Functions

- static const char * getColorForDuration (const float duration)
- static std::string formatLogMessage (LogLevel level, const std::string &message)

Static Private Attributes

- static constexpr std::array< const char *, 4 > LOG_LEVEL_COLOR
- static constexpr std::array< const char *, 2 > LOG_LEVEL_STRING = {"INFO", "WARNING"}

6.7.1 Detailed Description

Definition at line 17 of file Logger.hpp.

6.7.2 Member Enumeration Documentation

6.7.2.1 ColorIndex

enum utl::Logger::ColorIndex : uint8_t [private]

Enumerator

COLOR_ERROR	
COLOR_INFO	
COLOR_WARNING	
COLOR_RESET	

Definition at line 47 of file Logger.hpp.

6.7.3 Constructor & Destructor Documentation

6.7.3.1 Logger() [1/3]

utl::Logger::Logger (
const Logger &) [delete]

6.7.3.2 Logger() [2/3]

utl::Logger::Logger (
Logger &&) [delete]

6.7.3.3 Logger() [3/3]

utl::Logger::Logger () [private], [default]

6.7.3.4 ~Logger()

utl::Logger::~Logger () [private], [default]

6.7.4 Member Function Documentation

6.7.4.1 formatLogMessage()

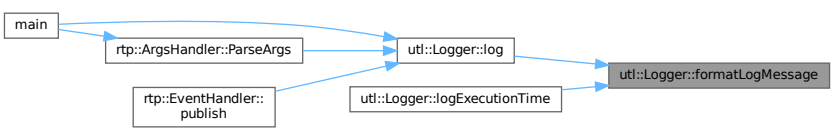
static std::string utl::Logger::formatLogMessage (
LogLevel level,
const std::string & message) [inline], [static], [nodiscard], [private]

Definition at line 74 of file Logger.hpp.

References LOG_LEVEL_STRING.

Referenced by log(), and logExecutionTime().

Here is the caller graph for this function:



6.7.4.2 getColorForDuration()

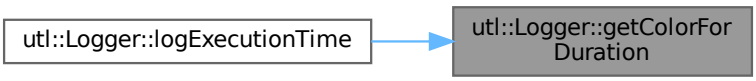
static const char * utl::Logger::getColorForDuration (
const float duration) [inline], [static], [nodiscard], [private]

Definition at line 67 of file Logger.hpp.

References COLOR_ERROR, COLOR_INFO, COLOR_WARNING, and LOG_LEVEL_COLOR.

Referenced by logExecutionTime().

Here is the caller graph for this function:



6.7.4.3 init()

void utl::Logger::init () [static]

Definition at line 7 of file logger.cpp.

Referenced by main().

Here is the caller graph for this function:

6.7.4.4 log()

static void utl::Logger::log (
const std::string & message,
const LogLevel & logLevel) [inline], [static]

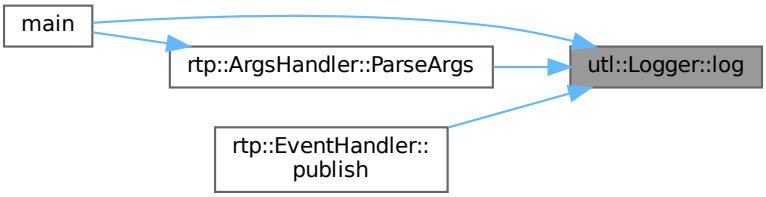
Definition at line 40 of file Logger.hpp.

References COLOR_INFO, COLOR_RESET, COLOR_WARNING, formatLogMessage(), utl::INFO, and LOG_LEVEL_COLOR.

Referenced by main(), rtp::ArgsHandler::ParseArgs(), and rtp::EventHandler::publish().

Here is the call graph for this function:

Here is the caller graph for this function:



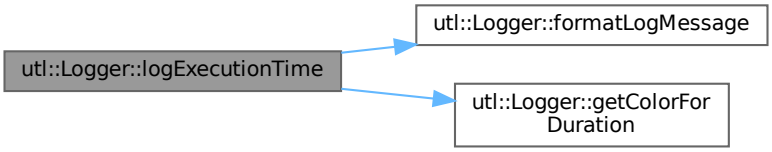
6.7.4.5 logExecutionTime()

template<typename Func >
static void utl::Logger::logExecutionTime (
const std::string & message,
Func && func) [inline], [static]

Definition at line 28 of file Logger.hpp.

References COLOR_RESET, formatLogMessage(), getColorForDuration(), utl::INFO, and LOG_LEVEL_COLOR.

Here is the call graph for this function:



6.7.4.6 operator=() [1/2]

Logger & utl::Logger::operator= (
const Logger &) [delete]

6.7.4.7 operator=() [2/2]

Logger & utl::Logger::operator= (
Logger &&) [delete]

6.7.5 Member Data Documentation

6.7.5.1 LOG_LEVEL_COLOR

std::array<const char *, 4> utl::Logger::LOG_LEVEL_COLOR [static], [constexpr], [private]

Initial value:
= {
"\033[31m",
"\033[32m",
"\033[33m",
"\033[0m\n"
}

Definition at line 55 of file Logger.hpp.

Referenced by getColorForDuration(), log(), and logExecutionTime().

6.7.5.2 LOG_LEVEL_STRING

std::array<const char *, 2> utl::Logger::LOG_LEVEL_STRING = {"INFO", "WARNING"} [static], [constexpr], [private]

Definition at line 62 of file Logger.hpp.

Referenced by formatLogMessage().

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

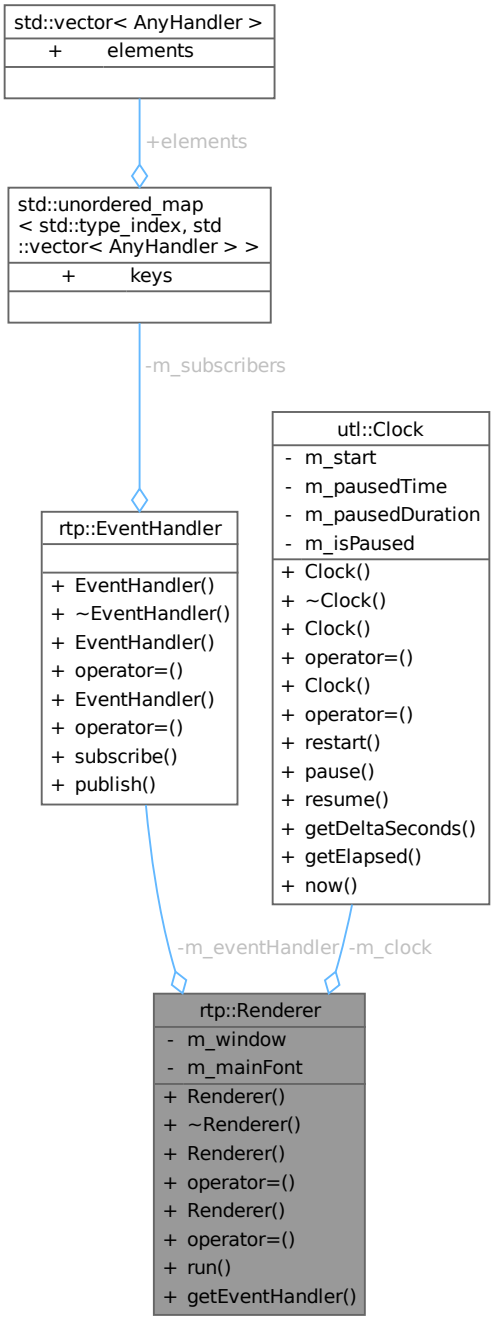
- /home/masina/Projects/Epitech/rtype/modules/Utils/include/Utils/Logger.hpp
- /home/masina/Projects/Epitech/rtype/modules/Utils/src/logger.cpp

6.8 rtp::Renderer Class Reference

Class for the renderer.

#include <Renderer.hpp>

Collaboration diagram for rtp::Renderer:



Public Member Functions

- Renderer (unsigned int height, unsigned int width, unsigned int frameLimit)
- ~Renderer ()=default
- Renderer (const Renderer &)=delete
- Renderer & operator= (const Renderer &)=delete
- Renderer (Renderer &&)=delete
- Renderer & operator= (Renderer &&)=delete
- void run ()
- EventHandler & getEventHandler ()

Private Attributes

- sf::RenderWindow m_window
- sf::Font m_mainFont
- EventHandler m_eventHandler
- utl::Clock m_clock

6.8.1 Detailed Description

Class for the renderer.

Definition at line 28 of file Renderer.hpp.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Renderer() [1/3]

```
rtp::Renderer::Renderer (
            unsigned int height,
            unsigned int width,
            unsigned int frameLimit)
```

Definition at line 6 of file renderer.cpp.

6.8.2.2 ~Renderer()

```
rtp::Renderer::~Renderer ()  [default]
```

6.8.2.3 Renderer() [2/3]

```
rtp::Renderer::Renderer (
            const Renderer & )  [delete]
```

6.8.2.4 Renderer() [3/3]

```
rtp::Renderer::Renderer (
            Renderer && )  [delete]
```

6.8.3 Member Function Documentation

6.8.3.1 getEventHandler()

EventHandler & rtp::Renderer::getEventHandler () [inline]

Definition at line 42 of file Renderer.hpp.

References m_eventHandler.

6.8.3.2 operator=() [1/2]

Renderer & rtp::Renderer::operator= (
const Renderer &) [delete]

6.8.3.3 operator=() [2/2]

Renderer & rtp::Renderer::operator= (
Renderer &&) [delete]

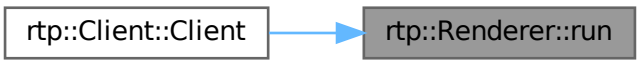
6.8.3.4 run()

void rtp::Renderer::run ()

Definition at line 27 of file renderer.cpp.

Referenced by rtp::Client::Client().

Here is the caller graph for this function:



6.8.4 Member Data Documentation

6.8.4.1 m_clock

utl::Clock rtp::Renderer::m_clock [private]

Definition at line 48 of file Renderer.hpp.

6.8.4.2 m_eventHandler

EventHandler rtp::Renderer::m_eventHandler [private]

Definition at line 47 of file Renderer.hpp.

Referenced by getEventHandler().

6.8.4.3 m_mainFont

sf::Font rtp::Renderer::m_mainFont [private]

Definition at line 46 of file Renderer.hpp.

6.8.4.4 m_window

sf::RenderWindow rtp::Renderer::m_window [private]

Definition at line 45 of file Renderer.hpp.

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

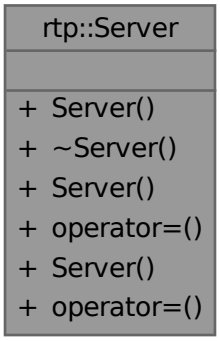
- /home/masina/Projects/Epitech/rtype/client/include/R-Type/Renderer/Renderer.hpp
- /home/masina/Projects/Epitech/rtype/client/src/renderer/renderer.cpp

6.9 rtp::Server Class Reference

Class for the server.

#include <Server.hpp>

Collaboration diagram for rtp::Server:



Public Member Functions

- Server (const ArgsConfig &config)
- ~Server ()=default
- Server (const Server &)=delete
- Server & operator= (const Server &)=delete
- Server (Server &&)=delete
- Server & operator= (Server &&)=delete

6.9.1 Detailed Description

Class for the server.

Definition at line 17 of file Server.hpp.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Server() [1/3]

rtp::Server::Server (
const ArgsConfig & config) [inline], [explicit]

Definition at line 21 of file Server.hpp.

6.9.2.2 ~Server()

rtp::Server::~Server () [default]

6.9.2.3 Server() [2/3]

rtp::Server::Server (
const Server &) [delete]

6.9.2.4 Server() [3/3]

rtp::Server::Server (
Server &&) [delete]

6.9.3 Member Function Documentation

6.9.3.1 operator=() [1/2]

Server & rtp::Server::operator= (
const Server &) [delete]

6.9.3.2 operator=() [2/2]

Server & rtp::Server::operator= (
Server &&) [delete]

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

- /home/masina/Projects/Epitech/rtype/server/include/R-Type/Server.hpp

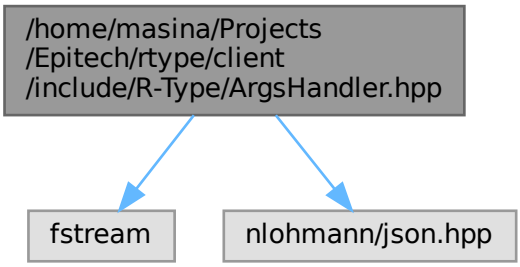
Chapter 7

File Documentation

7.1 /home/masina/Projects/Epitech/rtype/client/include/R-Type/↩ ArgsHandler.hpp File Reference

This file contains the ArgsHandler class declaration.

#include <fstream>
#include "nlohmann/json.hpp"
Include dependency graph for ArgsHandler.hpp:



Classes

- struct rtp::ArgsConfig
- struct rtp::EnvConfig
- class rtp::ArgsHandler

  Class to handle command line arguments.

Namespaces

- namespace rtp

Typedefs

- using rtp::json = nlohmann::json

7.1.1 Detailed Description

This file contains the ArgsHandler class declaration.

Definition in file ArgsHandler.hpp.

7.2 ArgsHandler.hpp

Go to the documentation of this file.

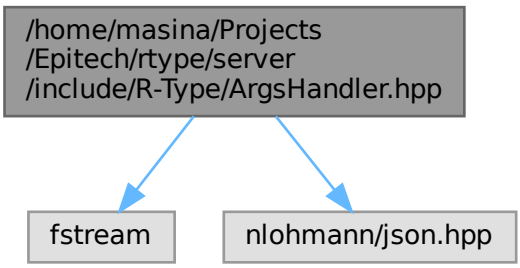
```
00001 ///
00002 /// @file ArgsHandler.hpp
00003 /// @brief This file contains the ArgsHandler class declaration
00004 /// @namespace rtp
00005 ///
00006
00007 #pragma once
00008
00009 #include <fstream>
00010
00011 #include "nlohmann/json.hpp"
00012
00013 namespace rtp
00014 {
00015
00016     using json = nlohmann::json;
00017
00018     struct ArgsConfig
00019     {
00020         unsigned int width = 960;
00021         unsigned int height = 540;
00022         unsigned int frameLimit = 240;
00023
00024         static ArgsConfig fromFile(const std::string &path)
00025         {
00026             ArgsConfig cfg;
00027             std::ifstream file(path);
00028             if (!file.is_open())
00029             {
00030                 throw std::runtime_error("Cannot open config file: " + path);
00031             }
00032
00033             json j;
00034             file » j;
00035
00036             if (j.contains("window"))
00037             {
00038                 const auto &w = j["window"];
00039                 if (w.contains("width"))
00040                     cfg.width = w["width"];
00041                 if (w.contains("height"))
00042                     cfg.height = w["height"];
00043                 if (w.contains("frame_limit"))
00044                     cfg.frameLimit = w["frame_limit"];
00045             }
00046             return cfg;
00047         }
00048     }; // struct Config
00049     struct EnvConfig
00050     {
00051     };
00052
00053     ///
00054     /// @class ArgsHandler
00055     /// @brief Class to handle command line arguments
00056     /// @namespace rtp
00057     ///
00058     class ArgsHandler
00059     {
00060
00061         public:
00062             ArgsHandler() = default;
```

```
00063             ~ArgsHandler() = default;
00064
00065             ArgsHandler(const ArgsHandler &) = delete;
00066             ArgsHandler &operator=(const ArgsHandler &) = delete;
00067             ArgsHandler(ArgsHandler &&) = delete;
00068             ArgsHandler &operator=(ArgsHandler &&) = delete;
00069
00070             static ArgsConfig ParseArgs(int argc, const char *const argv[]);
00071             static EnvConfig ParseEnv(const char *const env[]);
00072
00073         private:
00074     }; // class ArgsHandler
00075
00076 } // namespace rtp
```

7.3 /home/masina/Projects/Epitech/rtype/server/include/R-Type/ArgsHandler.hpp File Reference

This file contains the ArgsHandler class declaration.

#include <fstream>
#include "nlohmann/json.hpp"
Include dependency graph for ArgsHandler.hpp:



Classes

- struct rtp::ArgsConfig
- struct rtp::EnvConfig
- class rtp::ArgsHandler

  Class to handle command line arguments.

Namespaces

- namespace rtp

7.3.1 Detailed Description

This file contains the ArgsHandler class declaration.

Definition in file ArgsHandler.hpp.

7.4 ArgsHandler.hpp

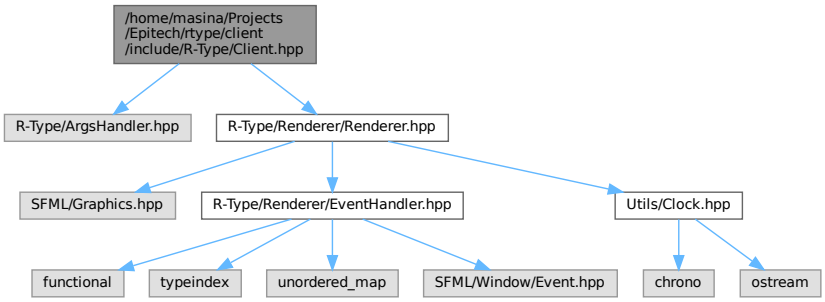
Go to the documentation of this file.

```
00001 ///
00002 /// @file ArgsHandler.hpp
00003 /// @brief This file contains the ArgsHandler class declaration
00004 /// @namespace rtp
00005 ///
00006
00007 #pragma once
00008
00009 #include <fstream>
00010
00011 #include "nlohmann/json.hpp"
00012
00013 namespace rtp
00014 {
00015
00016     using json = nlohmann::json;
00017
00018     struct ArgsConfig
00019     {
00020         std::string host = "0.0.0.0";
00021         unsigned int port = 2560;
00022
00023         static ArgsConfig fromFile(const std::string &path)
00024         {
00025             ArgsConfig cfg;
00026             std::ifstream file(path);
00027             if (!file.is_open())
00028             {
00029                 throw std::runtime_error("Cannot open config file: " + path);
00030             }
00031
00032             json j;
00033             file » j;
00034
00035             if (j.contains("host"))
00036                 cfg.host = j["host"];
00037             if (j.contains("port"))
00038                 cfg.port = j["port"];
00039             return cfg;
00040         }
00041     }; // struct Config
00042     struct EnvConfig
00043     {
00044     };
00045
00046     ///
00047     /// @class ArgsHandler
00048     /// @brief Class to handle command line arguments
00049     /// @namespace rtp
00050     ///
00051     class ArgsHandler
00052     {
00053
00054         public:
00055             ArgsHandler() = default;
00056             ~ArgsHandler() = default;
00057
00058             ArgsHandler(const ArgsHandler &) = delete;
00059             ArgsHandler &operator=(const ArgsHandler &) = delete;
00060             ArgsHandler(ArgsHandler &&) = delete;
00061             ArgsHandler &operator=(ArgsHandler &&) = delete;
00062
00063             static ArgsConfig ParseArgs(int argc, const char *const argv[]);
00064             static EnvConfig ParseEnv(const char *const env[]);
00065
00066         private:
00067     }; // class ArgsHandler
00068
00069 } // namespace rtp
```

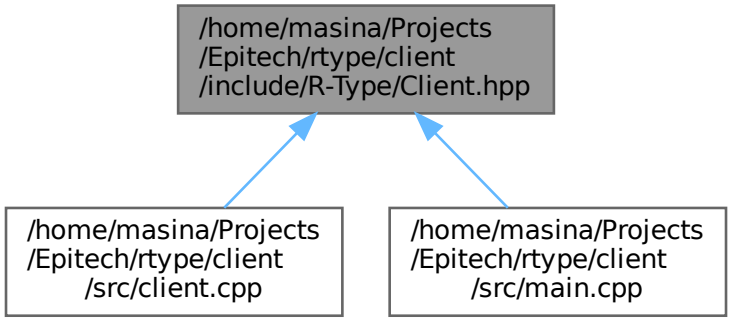
7.5 /home/masina/Projects/Epitech/rtype/client/include/R-Type/Client.hpp File Reference

This file contains the Client class declaration.

#include ”R-Type/ArgsHandler.hpp”
#include ”R-Type/Renderer/Renderer.hpp”
Include dependency graph for Client.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class rtp::Client

  Class for the client.

Namespaces

- namespace rtp

7.5.1 Detailed Description

This file contains the Client class declaration.

Definition in file Client.hpp.

7.6 Client.hpp

Go to the documentation of this file.

```
00001 ///
00002 /// @file Client.hpp
00003 /// @brief This file contains the Client class declaration
00004 /// @namespace rtp
00005 ///
00006
00007 #pragma once
00008
00009 #include "R-Type/ArgsHandler.hpp"
00010 #include "R-Type/Renderer/Renderer.hpp"
00011
00012 namespace rtp
00013 {
00014
00015     ///
00016     /// @class Client
00017     /// @brief Class for the client
00018     /// @namespace rtp
00019     ///
00020     class Client
00021     {
00022
00023         public:
00024             explicit Client(ArgsConfig cfg);
00025             ~Client() = default;
00026
00027             Client(const Client &) = delete;
00028             Client &operator=(const Client &) = delete;
00029             Client(Client &&) = delete;
00030             Client &operator=(Client &&) = delete;
00031
00032         private:
00033             Renderer m_renderer;
00034     }; // class Client
00035
00036 } // namespace rtp
```

7.7 /home/masina/Projects/Epitech/rtype/client/include/R-Type/Generated/Version.hpp File Reference

Macros

- #define PROJECT_NAME "r-type_client"
- #define PROJECT_VERSION "0.0.0"
- #define PROJECT_VERSION_MAJOR "0"
- #define PROJECT_VERSION_MINOR "0"
- #define PROJECT_VERSION_PATCH "0"
- #define GIT_COMMIT_HASH "76c28e2"
- #define GIT_TAG "76c28e2"
- #define BUILD_TYPE "Release"

7.7.1 Macro Definition Documentation

7.7.1.1 BUILD_TYPE

#define BUILD_TYPE "Release"

Definition at line 15 of file Version.hpp.

Referenced by main().

7.7.1.2 GIT_COMMIT_HASH

#define GIT_COMMIT_HASH "76c28e2"

Definition at line 13 of file Version.hpp.

Referenced by main().

7.7.1.3 GIT_TAG

#define GIT_TAG "76c28e2"

Definition at line 14 of file Version.hpp.

Referenced by main().

7.7.1.4 PROJECT_NAME

#define PROJECT_NAME "r-type_client"

Definition at line 7 of file Version.hpp.

Referenced by main().

7.7.1.5 PROJECT_VERSION

#define PROJECT_VERSION "0.0.0"

Definition at line 8 of file Version.hpp.

Referenced by main().

7.7.1.6 PROJECT_VERSION_MAJOR

#define PROJECT_VERSION_MAJOR "0"

Definition at line 9 of file Version.hpp.

7.7.1.7 PROJECT_VERSION_MINOR

#define PROJECT_VERSION_MINOR "0"

Definition at line 10 of file Version.hpp.

7.7.1.8 PROJECT_VERSION_PATCH

#define PROJECT_VERSION_PATCH "0"

Definition at line 11 of file Version.hpp.

7.8 Version.hpp

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 //
      ================================================================================
00004 // DO NOT EDIT THIS FILE MANUALLY. IT IS GENERATED BY CMAKE DURING THE BUILD PROCESS.
00005 //
      ================================================================================
00006
00007 #define PROJECT_NAME "r-type_client"
00008 #define PROJECT_VERSION "0.0.0"
00009 #define PROJECT_VERSION_MAJOR "0"
00010 #define PROJECT_VERSION_MINOR "0"
00011 #define PROJECT_VERSION_PATCH "0"
00012
00013 #define GIT_COMMIT_HASH "76c28e2"
00014 #define GIT_TAG "76c28e2"
00015 #define BUILD_TYPE "Release"
```

7.9 /home/masina/Projects/Epitech/rtype/server/include/R-Type/↩ Generated/Version.hpp File Reference

Macros

- #define PROJECT_NAME "r-type_server"
- #define PROJECT_VERSION "0.0.0"
- #define PROJECT_VERSION_MAJOR "0"
- #define PROJECT_VERSION_MINOR "0"
- #define PROJECT_VERSION_PATCH "0"
- #define GIT_COMMIT_HASH "76c28e2"
- #define GIT_TAG "76c28e2"
- #define BUILD_TYPE "Release"

7.9.1 Macro Definition Documentation

7.9.1.1 BUILD_TYPE

#define BUILD_TYPE "Release"

Definition at line 15 of file Version.hpp.

7.9.1.2 GIT_COMMIT_HASH

#define GIT_COMMIT_HASH "76c28e2"

Definition at line 13 of file Version.hpp.

7.9.1.3 GIT_TAG

#define GIT_TAG "76c28e2"

Definition at line 14 of file Version.hpp.

7.9.1.4 PROJECT_NAME

#define PROJECT_NAME "r-type_server"

Definition at line 7 of file Version.hpp.

7.9.1.5 PROJECT_VERSION

#define PROJECT_VERSION "0.0.0"

Definition at line 8 of file Version.hpp.

7.9.1.6 PROJECT_VERSION_MAJOR

#define PROJECT_VERSION_MAJOR "0"

Definition at line 9 of file Version.hpp.

7.9.1.7 PROJECT_VERSION_MINOR

#define PROJECT_VERSION_MINOR "0"

Definition at line 10 of file Version.hpp.

7.9.1.8 PROJECT_VERSION_PATCH

#define PROJECT_VERSION_PATCH "0"

Definition at line 11 of file Version.hpp.

7.10 Version.hpp

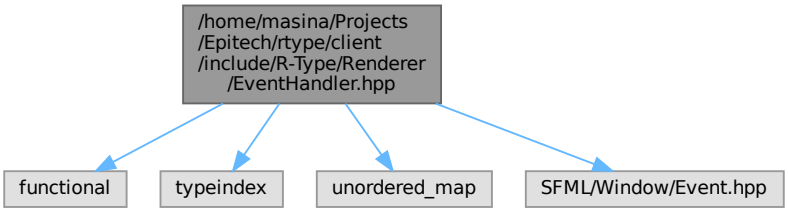
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 //
      ================================================================================
00004 // DO NOT EDIT THIS FILE MANUALLY. IT IS GENERATED BY CMAKE DURING THE BUILD PROCESS.
00005 //
      ================================================================================
00006
00007 #define PROJECT_NAME "r-type_server"
00008 #define PROJECT_VERSION "0.0.0"
00009 #define PROJECT_VERSION_MAJOR "0"
00010 #define PROJECT_VERSION_MINOR "0"
00011 #define PROJECT_VERSION_PATCH "0"
00012
00013 #define GIT_COMMIT_HASH "76c28e2"
00014 #define GIT_TAG "76c28e2"
00015 #define BUILD_TYPE "Release"
```

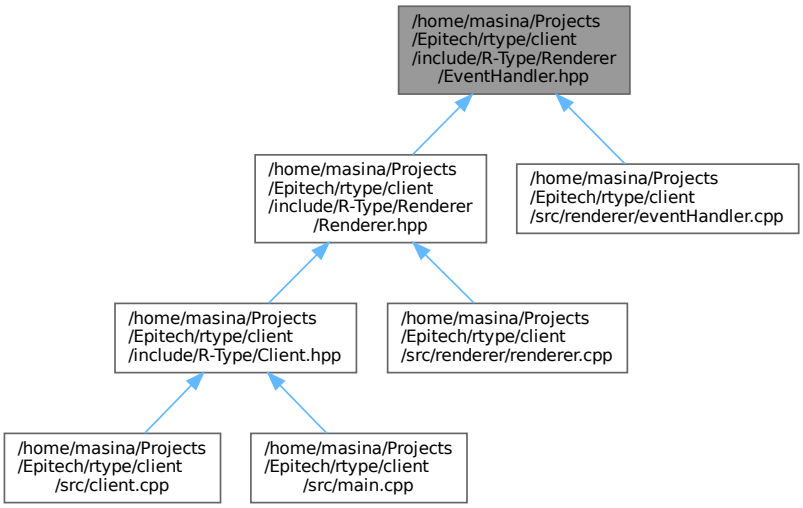
7.11 /home/masina/Projects/Epitech/rtype/client/include/R-Type/Renderer/EventHandler.hpp File Reference

This file contains the EventHandler class declaration.

#include <functional>
#include <typeindex>
#include <unordered_map>
#include <SFML/Window/Event.hpp>
Include dependency graph for EventHandler.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class rtp::EventHandler

  Class for the EventHandler.

Namespaces

- namespace rtp

7.11.1 Detailed Description

This file contains the EventHandler class declaration.

Definition in file EventHandler.hpp.

7.12 EventHandler.hpp

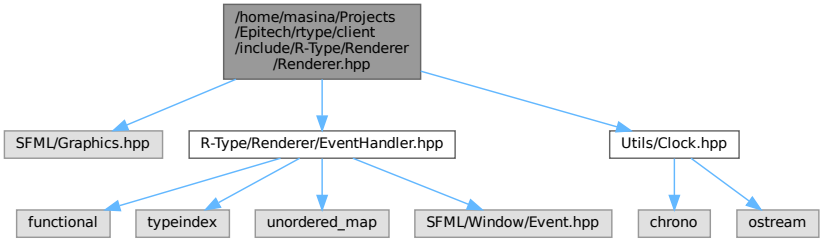
Go to the documentation of this file.

```
00001 ///
00002 /// @file EventHandler.hpp
00003 /// @brief This file contains the EventHandler class declaration
00004 /// @namespace rtp
00005 ///
00006 #pragma once
00007
00008 #include <functional>
00009 #include <typeindex>
00010 #include <unordered_map>
00011
00012 #include <SFML/Window/Event.hpp>
00013
00014 namespace rtp
00015 {
00016     ///
00017     /// @class EventHandler
00018     /// @brief Class for the EventHandler
00019     /// @namespace rtp
00020     ///
00021     class EventHandler
00022     {
00023
00024         public:
00025             template <typename T> using Handler = std::function<void(const T &)>;
00026
00027             EventHandler() = default;
00028             ~EventHandler() = default;
00029
00030             EventHandler(const EventHandler &) = delete;
00031             EventHandler &operator=(const EventHandler &) = delete;
00032             EventHandler(EventHandler &&) = delete;
00033             EventHandler &operator=(EventHandler &&) = delete;
00034
00035             template <typename T> void subscribe(Handler<T> handler)
00036             {
00037                 auto &handlers = m_subscribers[std::type_index(typeid(T))];
00038                 handlers.push_back(
00039                     [h = std::move(handler)](const sf::Event &e)
00040                     {
00041                         if (const auto *data = e.getIf<T>())
00042                         {
00043                             h(*data);
00044                         }
00045                     });
00046             }
00047
00048             void publish(const sf::Event &e);
00049
00050         private:
00051             using AnyHandler = std::function<void(const sf::Event &)>;
00052             std::unordered_map<std::type_index, std::vector<AnyHandler>> m_subscribers;
00053
00054     }; // class EventHandler
00055
00056 } // namespace rtp
```

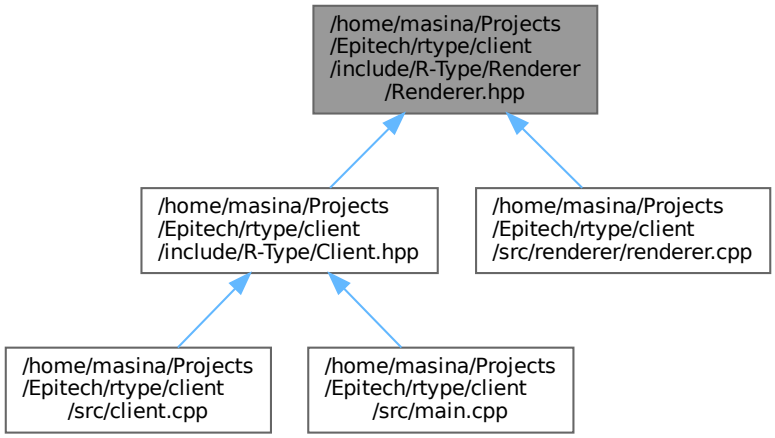
7.13 /home/masina/Projects/Epitech/rtype/client/include/R-Type/Renderer/Renderer.hpp File Reference

This file contains the Renderer class declaration.

#include <SFML/Graphics.hpp>
#include "R-Type/Renderer/EventHandler.hpp"
#include "Utils/Clock.hpp"
Include dependency graph for Renderer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class rtp::Renderer
  Class for the renderer.

Namespaces

- namespace rtp
- namespace sf

7.13.1 Detailed Description

This file contains the Renderer class declaration.

Definition in file Renderer.hpp.

7.14 Renderer.hpp

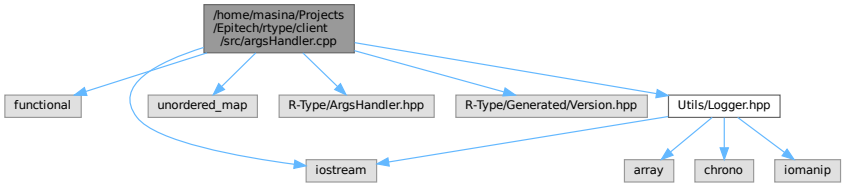
Go to the documentation of this file.

```
00001 ///
00002 /// @file Renderer.hpp
00003 /// @brief This file contains the Renderer class declaration
00004 /// @namespace rtp
00005 ///
00006
00007 #pragma once
00008
00009 #include <SFML/Graphics.hpp>
00010
00011 #include "R-Type/Renderer/EventHandler.hpp"
00012 #include "Utils/Clock.hpp"
00013
00014 // Forward declaration
00015 namespace sf
00016 {
00017     class RenderWindow;
00018 }
00019
00020 namespace rtp
00021 {
00022
00023     ///
00024     /// @class Renderer
00025     /// @brief Class for the renderer
00026     /// @namespace rtp
00027     ///
00028     class Renderer
00029     {
00030
00031         public:
00032             Renderer(unsigned int height, unsigned int width, unsigned int frameLimit);
00033             ~Renderer() = default;
00034
00035             Renderer(const Renderer &) = delete;
00036             Renderer &operator=(const Renderer &) = delete;
00037             Renderer(Renderer &&) = delete;
00038             Renderer &operator=(Renderer &&) = delete;
00039
00040             void run();
00041
00042             EventHandler &getEventHandler() { return m_eventHandler; }
00043
00044         private:
00045             sf::RenderWindow m_window;
00046             sf::Font m_mainFont;
00047             EventHandler m_eventHandler;
00048             utl::Clock m_clock;
00049     }; // class Renderer
00050
00051 } // namespace rtp
```

7.15 /home/masina/Projects/Epitech/rtype/client/src/argsHandler.cpp File Reference

```
#include <functional>
#include <iostream>
#include <unordered_map>
#include "R-Type/ArgsHandler.hpp"
#include "R-Type/Generated/Version.hpp"
#include "Utils/Logger.hpp"
```

Include dependency graph for argsHandler.cpp:



Macros

- #define APP_EXTENSION ""

Variables

- static constexpr std::string_view HELP_MESSAGE
- static constexpr std::string_view VERSION_MESSAGE

7.15.1 Macro Definition Documentation

7.15.1.1 APP_EXTENSION

#define APP_EXTENSION ""

Definition at line 8 of file argsHandler.cpp.

7.15.2 Variable Documentation

7.15.2.1 HELP_MESSAGE

std::string_view HELP_MESSAGE [static], [constexpr]

Initial value:
```
= "Usage: " PROJECT_NAME APP_EXTENSION " [options]\n\n"
                                       "Options:\n"
                                       "\t--help, -h       Show this help message\n"
                                       "\t--version, -v    Show version information\n"
                                       "\t--config, -c     Specify path to config file\n"
```

Definition at line 15 of file argsHandler.cpp.

Referenced by rtp::ArgsHandler::ParseArgs().

7.15.2.2 VERSION_MESSAGE

std::string_view VERSION_MESSAGE [static], [constexpr]

Initial value:
```
= PROJECT_NAME " version " PROJECT_VERSION "\n"
                                                    "Build type: " BUILD_TYPE "\n"
                                                    "Git tag: " GIT_TAG "\n"
                                                    "Git commit hash: " GIT_COMMIT_HASH "\n"
```

Definition at line 20 of file argsHandler.cpp.

Referenced by rtp::ArgsHandler::ParseArgs().

7.16 argsHandler.cpp

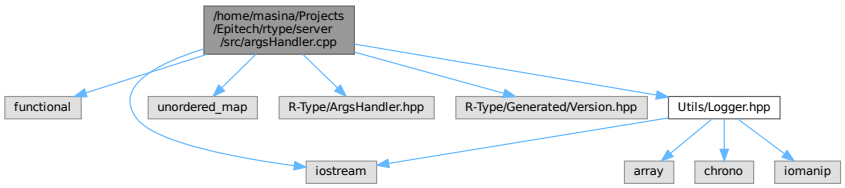
Go to the documentation of this file.

```
00001 #include <functional>
00002 #include <iostream>
00003 #include <unordered_map>
00004
00005 #ifdef _WIN32
00006 #define APP_EXTENSION ".exe"
00007 #else
00008 #define APP_EXTENSION ""
00009 #endif
00010
00011 #include "R-Type/ArgsHandler.hpp"
00012 #include "R-Type/Generated/Version.hpp"
00013 #include "Utils/Logger.hpp"
00014
00015 static constexpr std::string_view HELP_MESSAGE = "Usage: " PROJECT_NAME APP_EXTENSION " [options]\n\n"
00016                                                  "Options:\n"
00017                                                  "\t--help, -h       Show this help message\n"
00018                                                  "\t--version, -v    Show version information\n"
00019                                                  "\t--config, -c     Specify path to config file\n";
00020 static constexpr std::string_view VERSION_MESSAGE = PROJECT_NAME " version " PROJECT_VERSION "\n"
00021                                                      "Build type: " BUILD_TYPE "\n"
00022                                                      "Git tag: " GIT_TAG "\n"
00023                                                      "Git commit hash: " GIT_COMMIT_HASH "\n";
00024
00025 rtp::ArgsConfig rtp::ArgsHandler::ParseArgs(const int argc, const char *const argv[])
00026 {
00027     if (argc <= 1)
00028         return {};
00029
00030     using ArgHandler = std::function<void(const char *arg)>;
00031     std::unordered_map<std::string_view, ArgHandler> handlers;
00032     ArgsConfig config{};
00033     for (const auto opt : {"-h", "--help"})
00034         handlers[opt] = [](const char *) { std::cout << HELP_MESSAGE; };
00035     for (const auto opt : {"-v", "--version"})
00036         handlers[opt] = [](const char *) { std::cout << VERSION_MESSAGE; };
00037
00038     for (const auto opt : {"-c", "--config"})
00039         handlers[opt] = [&config](const char *arg)
00040         {
00041             if (!arg)
00042                 throw std::runtime_error("Missing config file argument");
00043             config = ArgsConfig::fromFile(arg);
00044             utl::Logger::log("Loaded config from file: " + std::string(arg), utl::LogLevel::INFO);
00045             std::cout << "\tWidth: " << config.width << '\n'
00046                       << "\tHeight: " << config.height << '\n'
00047                       << "\tFrameLimit: " << config.frameLimit << "\n";
00048         };
00049
00050     const std::string_view key = argv[1];
00051     const char *argValue = (argc > 2) ? argv[2] : nullptr;
00052
00053     if (const auto it = handlers.find(key); it != handlers.end())
00054     {
00055         it->second(argValue);
00056         return config;
00057     }
00058
00059     throw std::runtime_error("Unknown argument: " + std::string(key));
00060 }
```

```
00061
00062 rtp::EnvConfig rtp::ArgsHandler::ParseEnv(const char *const env[])
00063 {
00064     (void)env; // Currently unused
00065     return {};
00066 }
```

7.17 /home/masina/Projects/Epitech/rtype/server/src/argsHandler.cpp File Reference

#include <functional>
#include <iostream>
#include <unordered_map>
#include "R-Type/ArgsHandler.hpp"
#include "R-Type/Generated/Version.hpp"
#include "Utils/Logger.hpp"
Include dependency graph for argsHandler.cpp:



Macros

- #define APP_EXTENSION ""

Variables

- static constexpr std::string_view HELP_MESSAGE
- static constexpr std::string_view VERSION_MESSAGE

7.17.1 Macro Definition Documentation

7.17.1.1 APP_EXTENSION

#define APP_EXTENSION ""

Definition at line 9 of file argsHandler.cpp.

7.17.2 Variable Documentation

7.17.2.1 HELP_MESSAGE

std::string_view HELP_MESSAGE [static], [constexpr]

Initial value:
```
= "Usage: " PROJECT_NAME APP_EXTENSION " [options]\n\n"
                                        "Options:\n"
                                        "\t--help, -h       Show this help message\n"
                                        "\t--version, -v    Show version information\n"
                                        "\t--config, -c     Specify path to config file\n"
```

Definition at line 16 of file argsHandler.cpp.

7.17.2.2 VERSION_MESSAGE

std::string_view VERSION_MESSAGE [static], [constexpr]

Initial value:
```
= PROJECT_NAME " version " PROJECT_VERSION "\n"
                                                "Build type: " BUILD_TYPE "\n"
                                                "Git tag: " GIT_TAG "\n"
                                                "Git commit hash: " GIT_COMMIT_HASH "\n"
```

Definition at line 21 of file argsHandler.cpp.

7.18 argsHandler.cpp

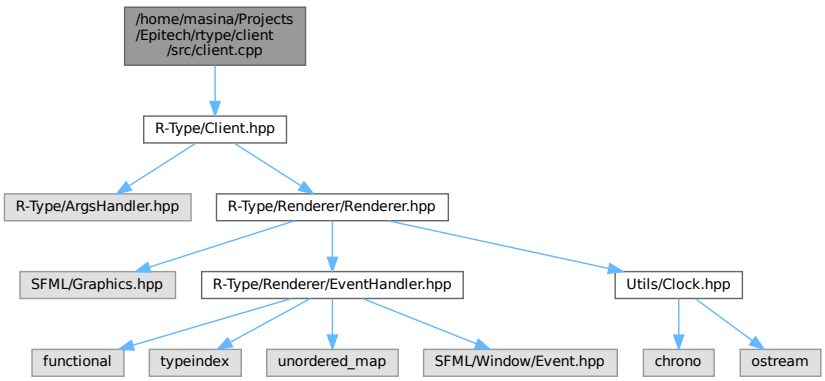
Go to the documentation of this file.

```
00001 #include <functional>
00002 #include <iostream>
00003 #include <unordered_map>
00004
00005 #ifdef _WIN32
00006 #include <windows.h>
00007 #define APP_EXTENSION ".exe"
00008 #else
00009 #define APP_EXTENSION ""
00010 #endif
00011
00012 #include "R-Type/ArgsHandler.hpp"
00013 #include "R-Type/Generated/Version.hpp"
00014 #include "Utils/Logger.hpp"
00015
00016 static constexpr std::string_view HELP_MESSAGE = "Usage: " PROJECT_NAME APP_EXTENSION " [options]\n\n"
00017                                                  "Options:\n"
00018                                                  "\t--help, -h       Show this help message\n"
00019                                                  "\t--version, -v    Show version information\n"
00020                                                  "\t--config, -c     Specify path to config file\n";
00021 static constexpr std::string_view VERSION_MESSAGE = PROJECT_NAME " version " PROJECT_VERSION "\n"
00022                                                     "Build type: " BUILD_TYPE "\n"
00023                                                     "Git tag: " GIT_TAG "\n"
00024                                                     "Git commit hash: " GIT_COMMIT_HASH "\n";
00025
00026 rtp::ArgsConfig rtp::ArgsHandler::ParseArgs(const int argc, const char *const argv[])
00027 {
00028     if (argc <= 1)
00029         return {};
00030
00031     using ArgHandler = std::function<void(const char *arg)>;
00032     std::unordered_map<std::string_view, ArgHandler> handlers;
00033     ArgsConfig config{};
00034     for (const auto opt : {"-h", "--help"})
00035         handlers[opt] = [](const char *) { std::cout << HELP_MESSAGE; };
00036     for (const auto opt : {"-v", "--version"})
00037         handlers[opt] = [](const char *) { std::cout << VERSION_MESSAGE; };
00038
00039     for (const auto opt : {"-c", "--config"})
00040         handlers[opt] = [&config](const char *arg)
```

```
00041         {
00042             if (!arg) {
00043                 throw std::runtime_error("Missing config file argument");
00044             }
00045             config = ArgsConfig::fromFile(arg);
00046             utl::Logger::log("Loaded config from file: " + std::string(arg), utl::LogLevel::INFO);
00047             std::cout « "\tHost: " « config.host « '\n' « "\tPort: " « config.port « '\n';
00048         };
00049
00050     const std::string_view key = argv[1];
00051     const char *argValue = (argc > 2) ? argv[2] : nullptr;
00052
00053     if (const auto it = handlers.find(key); it != handlers.end())
00054     {
00055         it->second(argValue);
00056         return config;
00057     }
00058
00059     throw std::runtime_error("Unknown argument: " + std::string(key));
00060 }
00061
00062 rtp::EnvConfig rtp::ArgsHandler::ParseEnv(const char *const env[])
00063 {
00064     (void)env; // Currently unused
00065     return {};
00066 }
```

7.19 /home/masina/Projects/Epitech/rtype/client/src/client.cpp File Reference

#include "R-Type/Client.hpp"
Include dependency graph for client.cpp:



7.20 client.cpp

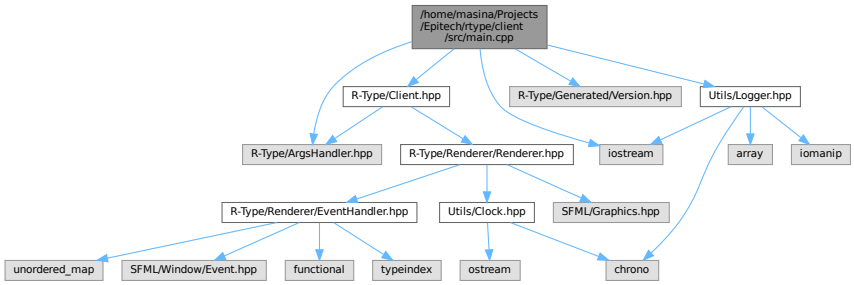
Go to the documentation of this file.

```
00001 #include "R-Type/Client.hpp"
00002
00003 rtp::Client::Client(const ArgsConfig cfg) : m_renderer(cfg.height, cfg.width, cfg.frameLimit) { m_renderer.run(); }
```

7.21 /home/masina/Projects/Epitech/rtype/client/src/main.cpp File Reference

#include <iostream>
#include "R-Type/ArgsHandler.hpp"

#include ”R-Type/Client.hpp”
#include ”R-Type/Generated/Version.hpp”
#include ”Utils/Logger.hpp”
Include dependency graph for main.cpp:



Functions

- int main (const int argc, const char ∗const argv[], const char ∗const env[])

7.21.1 Function Documentation

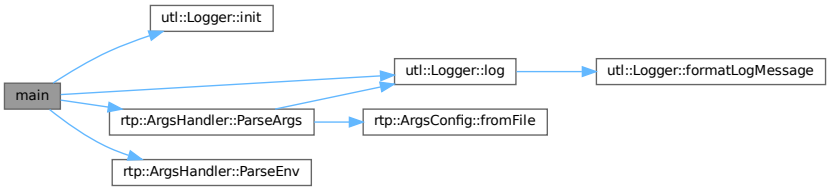
7.21.1.1 main()

int main (
const int argc,
const char ∗const argv[],
const char ∗const env[])

Definition at line 8 of file main.cpp.

References BUILD_TYPE, GIT_COMMIT_HASH, GIT_TAG, utl::INFO, utl::Logger::init(), utl::Logger::log(), rtp::ArgsHandler::ParseArgs(), rtp::ArgsHandler::ParseEnv(), PROJECT_NAME, PROJECT_VERSION, and utl::WARNING.

Here is the call graph for this function:



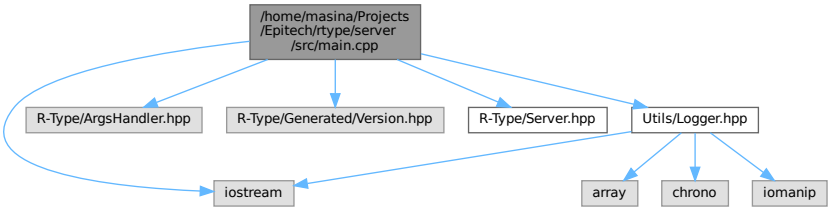
7.22 main.cpp

Go to the documentation of this file.

```
00001 #include <iostream>
00002
00003 #include "R-Type/ArgsHandler.hpp"
00004 #include "R-Type/Client.hpp"
00005 #include "R-Type/Generated/Version.hpp"
00006 #include "Utils/Logger.hpp"
00007
00008 int main(const int argc, const char *const argv[], const char *const env[])
00009 {
00010     utl::Logger::init();
00011     utl::Logger::log("args:", utl::LogLevel::INFO);
00012     for (int i = 0; i < argc; ++i)
00013     {
00014         std::cout << "\t[" << i << "] " << argv[i] << '\n';
00015     }
00016     utl::Logger::log("env:", utl::LogLevel::INFO);
00017     for (const char *const *e = env; *e != nullptr; ++e)
00018     {
00019         std::cout << "\t" << *e << '\n';
00020     }
00021     utl::Logger::log("PROJECT INFO:", utl::LogLevel::INFO);
00022     std::cout << "\tName: " PROJECT_NAME "\n"
00023                 "\tVersion: " PROJECT_VERSION "\n"
00024                 "\tBuild type: " BUILD_TYPE "\n"
00025                 "\tGit tag: " GIT_TAG "\n"
00026                 "\tGit commit hash: " GIT_COMMIT_HASH "\n";
00027
00028     try
00029     {
00030         const rtp::ArgsConfig argsConf = rtp::ArgsHandler::ParseArgs(argc, argv);
00031         const rtp::EnvConfig envConf = rtp::ArgsHandler::ParseEnv(env);
00032         rtp::Client client(argsConf);
00033     }
00034     catch (const std::exception &e)
00035     {
00036         utl::Logger::log(std::string("Exception: ") + e.what(), utl::LogLevel::WARNING);
00037         return EXIT_FAILURE;
00038     }
00039     catch (...)
00040     {
00041         utl::Logger::log("Unknown exception", utl::LogLevel::WARNING);
00042         return EXIT_FAILURE;
00043     }
00044     return EXIT_SUCCESS;
00045 }
```

7.23 /home/masina/Projects/Epitech/rtype/server/src/main.cpp File Reference

#include <iostream>
#include "R-Type/ArgsHandler.hpp"
#include "R-Type/Generated/Version.hpp"
#include "R-Type/Server.hpp"
#include "Utils/Logger.hpp"
Include dependency graph for main.cpp:



Functions

- int main (const int argc, const char *const argv[], const char *const env[])

7.23.1 Function Documentation

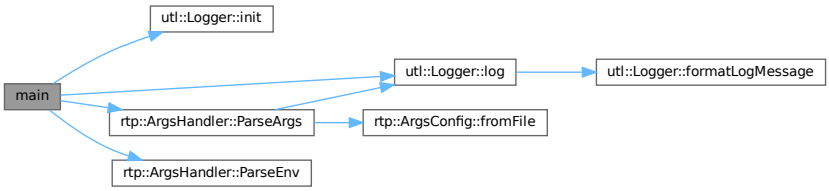
7.23.1.1 main()

int main (
const int argc,
const char *const argv[],
const char *const env[])

Definition at line 12 of file main.cpp.

References BUILD_TYPE, GIT_COMMIT_HASH, GIT_TAG, utl::INFO, utl::Logger::init(), utl::Logger::log(), rtp::ArgsHandler::ParseArgs(), rtp::ArgsHandler::ParseEnv(), PROJECT_NAME, PROJECT_VERSION, and utl::WARNING.

Here is the call graph for this function:



7.24 main.cpp

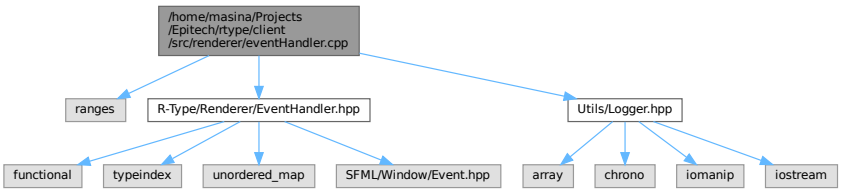
Go to the documentation of this file.

```
00001 #include <iostream>
00002
00003 #ifdef _WIN32
00004 #include <windows.h>
00005 #endif
00006
00007 #include "R-Type/ArgsHandler.hpp"
00008 #include "R-Type/Generated/Version.hpp"
00009 #include "R-Type/Server.hpp"
00010 #include "Utils/Logger.hpp"
00011
00012 int main(const int argc, const char *const argv[], const char *const env[])
00013 {
00014     utl::Logger::init();
00015     utl::Logger::log("args:", utl::LogLevel::INFO);
00016     for (int i = 0; i < argc; ++i)
00017     {
00018         std::cout << "\t[" << i << "] " << argv[i] << '\n';
00019     }
00020     utl::Logger::log("env:", utl::LogLevel::INFO);
00021     for (const char *const *e = env; *e != nullptr; ++e)
00022     {
00023         std::cout << "\t" << *e << '\n';
00024     }
00025     utl::Logger::log("PROJECT INFO:", utl::LogLevel::INFO);
00026     std::cout << "\tName: " PROJECT_NAME "\n"
00027                 "\tVersion: " PROJECT_VERSION "\n"
00028                 "\tBuild type: " BUILD_TYPE "\n"
```

```
00029                 "\tGit tag: " GIT_TAG "\n"
00030                 "\tGit commit hash: " GIT_COMMIT_HASH "\n";
00031
00032     try
00033     {
00034         const rtp::ArgsConfig argsConf = rtp::ArgsHandler::ParseArgs(argc, argv);
00035         const rtp::EnvConfig envConf = rtp::ArgsHandler::ParseEnv(env);
00036         rtp::Server server(argsConf);
00037     }
00038     catch (const std::exception &e)
00039     {
00040         utl::Logger::log(std::string("Exception: ") + e.what(), utl::LogLevel::WARNING);
00041         return EXIT_FAILURE;
00042     }
00043     catch (...)
00044     {
00045         utl::Logger::log("Unknown exception", utl::LogLevel::WARNING);
00046         return EXIT_FAILURE;
00047     }
00048     return EXIT_SUCCESS;
00049 }
```

7.25 /home/masina/Projects/Epitech/rtype/client/src/renderer/eventHandler.cpp File Reference

#include <ranges>
#include "R-Type/Renderer/EventHandler.hpp"
#include "Utils/Logger.hpp"
Include dependency graph for eventHandler.cpp:



7.26 eventHandler.cpp

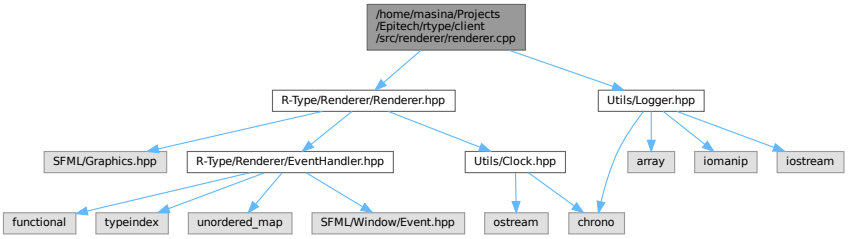
Go to the documentation of this file.

```
00001 #include <ranges>
00002
00003 #include "R-Type/Renderer/EventHandler.hpp"
00004 #include "Utils/Logger.hpp"
00005
00006 void rtp::EventHandler::publish(const sf::Event &e)
00007 {
00008     for (auto &val : m_subscribers | std::views::values)
00009     {
00010         std::erase_if(val,
00011                       [&](const AnyHandler &h)
00012                       {
00013                           try
00014                           {
00015                               h(e);
00016                               return false;
00017                           }
00018                           catch (const std::exception &ex)
00019                           {
00020                               utl::Logger::log("[EventBus] Handler threw exception: " + std::string(ex.what()),
00021                                                utl::LogLevel::WARNING);
00022                               return true;
00023                           }
00024                           catch (...)
```

```
00025                     {
00026                         utl::Logger::log("[EventBus] Handler threw unknown exception", utl::LogLevel::WARNING);
00027                         return true;
00028                     }
00029                 });
00030     }
00031 }
```

7.27 /home/masina/Projects/↩ Epitech/rtype/client/src/renderer/renderer.cpp File Reference

#include "R-Type/Renderer/Renderer.hpp"
#include "Utils/Logger.hpp"
Include dependency graph for renderer.cpp:



Variables

- static constexpr std::string_view WINDOW_TITLE = "R-Type - Client"

7.27.1 Variable Documentation

7.27.1.1 WINDOW_TITLE

std::string_view WINDOW_TITLE = "R-Type - Client" [static], [constexpr]

Definition at line 4 of file renderer.cpp.

7.28 renderer.cpp

Go to the documentation of this file.

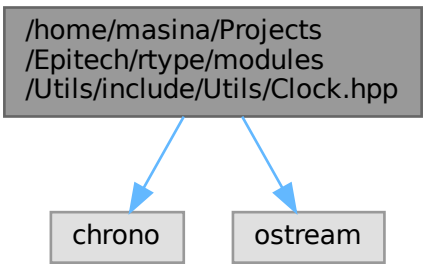
```
00001 #include "R-Type/Renderer/Renderer.hpp"
00002 #include "Utils/Logger.hpp"
00003
00004 static constexpr std::string_view WINDOW_TITLE = "R-Type - Client";
00005
00006 rtp::Renderer::Renderer(unsigned int height, unsigned int width, const unsigned int frameLimit)
00007     : m_window(sf::VideoMode({width, height}), std::string(WINDOW_TITLE))
00008 {
00009     m_window.setFramerateLimit(frameLimit);
00010     if (!m_mainFont.openFromFile("assets/fonts/r-type.otf"))
00011     {
00012         utl::Logger::log("Failed to load font", utl::LogLevel::WARNING);
00013     }
00014     m_eventHandler.subscribe<sf::Event::Closed>([&](const sf::Event::Closed &) { m_window.close(); });
```

```
00015     m_eventHandler.subscribe<sf::Event::KeyPressed>(
00016         [&](const sf::Event::KeyPressed &key)
00017         {
00018             utl::Logger::log(std::string("Key pressed: ") + std::to_string(static_cast<int>(key.scancode)),
00019                              utl::LogLevel::INFO);
00020             if (key.scancode == sf::Keyboard::Scancode::Escape)
00021             {
00022                 m_window.close();
00023             }
00024         });
00025 }
00026
00027 void rtp::Renderer::run()
00028 {
00029     sf::Text title(m_mainFont);
00030     sf::Text fpsText(m_mainFont);
00031     title.setString("RType Client");
00032     fpsText.setString("RType Client");
00033     title.setCharacterSize(50);
00034     fpsText.setCharacterSize(20);
00035     title.setFillColor(sf::Color::White);
00036     fpsText.setFillColor(sf::Color::White);
00037     title.setPosition({10.0F, 10.0F});
00038     fpsText.setPosition({10.0F, 70.0F});
00039     while (m_window.isOpen())
00040     {
00041         fpsText.setString("FPS " + std::to_string(static_cast<int>(1.0F / m_clock.getDeltaSeconds())));
00042         m_clock.restart();
00043         while (auto eventOpt = m_window.pollEvent())
00044         {
00045             m_eventHandler.publish(*eventOpt);
00046         }
00047         m_window.clear(sf::Color::Black);
00048         m_window.draw(title);
00049         m_window.draw(fpsText);
00050         m_window.display();
00051     }
00052 }
```

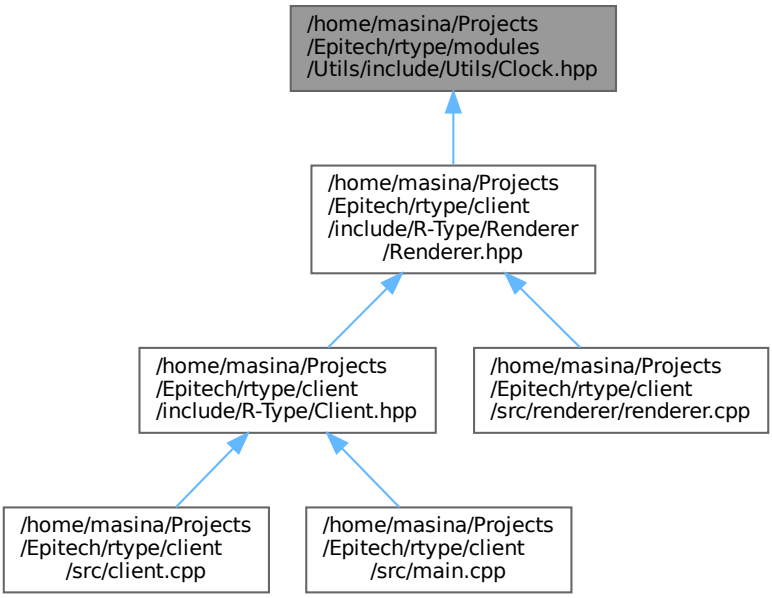
7.29 /home/masina/Projects/Epitech/rtype/modules/Utils/include/Utils/Clock.hpp File Reference

This file contains the Clock class.

#include <chrono>
#include <ostream>
Include dependency graph for Clock.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class utl::Clock

  Class for clock.

Namespaces

- namespace utl

7.29.1 Detailed Description

This file contains the Clock class.

Definition in file Clock.hpp.

7.30 Clock.hpp

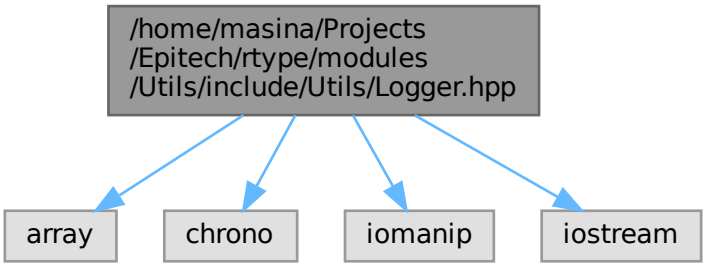
Go to the documentation of this file.

```
00001 ///
00002 /// @file Clock.hpp
00003 /// @brief This file contains the Clock class
00004 /// @namespace utl
00005 ///
00006
00007 #pragma once
00008
00009 #include <chrono>
```

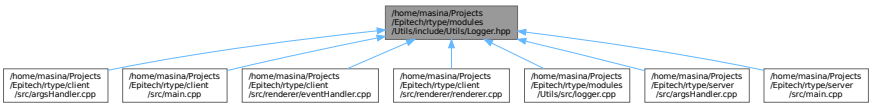
```
00010 #include <ostream>
00011
00012 namespace utl
00013 {
00014
00015     ///
00016     /// @class Clock
00017     /// @brief Class for clock
00018     /// @namespace utl
00019     ///
00020     class Clock
00021     {
00022
00023         public:
00024             using TimePoint = std::chrono::time_point<std::chrono::high_resolution_clock>;
00025
00026             explicit Clock(const bool startNow = true) : m_start{startNow ? now() : TimePoint()}, m_pausedDuration{0} {}
00027             ~Clock() = default;
00028
00029             Clock(const Clock &) = delete;
00030             Clock &operator=(const Clock &) = delete;
00031             Clock(Clock &&) = delete;
00032             Clock &operator=(Clock &&) = delete;
00033
00034             friend std::ostream &operator«(std::ostream &os, const Clock &clock)
00035             {
00036                 os « "Elapsed time: " « clock.getDeltaSeconds() « " seconds";
00037                 return os;
00038             }
00039
00040             static TimePoint now() { return std::chrono::high_resolution_clock::now(); }
00041             void restart()
00042             {
00043                 m_start = now();
00044                 m_pausedDuration = Duration(0);
00045                 m_isPaused = false;
00046             }
00047             void pause()
00048             {
00049                 if (!m_isPaused)
00050                 {
00051                     m_pausedTime = now();
00052                     m_isPaused = true;
00053                 }
00054             }
00055             void resume()
00056             {
00057                 if (m_isPaused)
00058                 {
00059                     m_pausedDuration += now() - m_pausedTime;
00060                     m_isPaused = false;
00061                 }
00062             }
00063             [[nodiscard]] float getDeltaSeconds() const
00064             {
00065                 if (m_isPaused)
00066                 {
00067                     return std::chrono::duration<float>(m_pausedTime - m_start - m_pausedDuration).count();
00068                 }
00069                 return std::chrono::duration<float>(now() - m_start - m_pausedDuration).count();
00070             }
00071
00072             template <typename Duration = std::chrono::seconds> [[nodiscard]] auto getElapsed() const
00073             {
00074                 return std::chrono::duration_cast<Duration>(now() - m_start - m_pausedDuration);
00075             }
00076
00077         private:
00078             using Duration = std::chrono::high_resolution_clock::duration;
00079
00080             TimePoint m_start;
00081             TimePoint m_pausedTime;
00082             Duration m_pausedDuration;
00083             bool m_isPaused{false};
00084
00085     }; // class Clock
00086
00087 } // namespace utl
```

7.31 /home/masina/Projects/Epitech/rtype/modules/Utils/include/↩ Utils/Logger.hpp File Reference

#include <array>
#include <chrono>
#include <iomanip>
#include <iostream>
Include dependency graph for Logger.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class utl::Logger

Namespaces

- namespace utl

Enumerations

- enum class utl::LogLevel : uint8_t { utl::INFO , utl::WARNING }

7.32 Logger.hpp

Go to the documentation of this file.

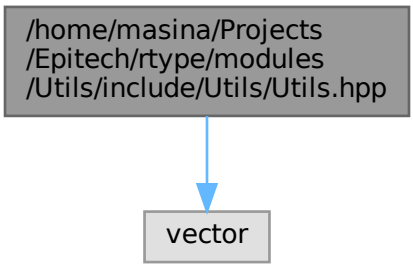
```
00001 #pragma once
00002
00003 #include <array>
00004 #include <chrono>
00005 #include <iomanip>
00006 #include <iostream>
00007
00008 namespace utl
00009 {
00010
00011     enum class LogLevel : uint8_t
00012     {
00013         INFO,
00014         WARNING
00015     };
00016
00017     class Logger
00018     {
00019
00020         public:
00021             Logger(const Logger &) = delete;
00022             Logger &operator=(const Logger &) = delete;
00023             Logger(Logger &&) = delete;
00024             Logger &operator=(Logger &&) = delete;
00025
00026             static void init();
00027
00028             template <typename Func> static void logExecutionTime(const std::string &message, Func &&func)
00029             {
00030                 const auto start = std::chrono::high_resolution_clock::now();
00031                 func();
00032                 const auto end = std::chrono::high_resolution_clock::now();
00033                 const auto duration = std::chrono::duration<float, std::milli>(end - start).count();
00034
00035                 std::cout « getColorForDuration(duration)
00036                         « formatLogMessage(LogLevel::INFO, message + " took " + std::to_string(duration) + " ms")
00037                         « LOG_LEVEL_COLOR[COLOR_RESET];
00038             }
00039
00040             static void log(const std::string &message, const LogLevel &logLevel)
00041             {
00042                 std::cout « (logLevel == LogLevel::INFO ? LOG_LEVEL_COLOR[COLOR_INFO] :
    LOG_LEVEL_COLOR[COLOR_WARNING])
00043                         « formatLogMessage(logLevel, message) « LOG_LEVEL_COLOR[COLOR_RESET];
00044             }
00045
00046         private:
00047             enum ColorIndex : uint8_t
00048             {
00049                 COLOR_ERROR,
00050                 COLOR_INFO,
00051                 COLOR_WARNING,
00052                 COLOR_RESET
00053             };
00054
00055             static constexpr std::array<const char *, 4> LOG_LEVEL_COLOR = {
00056                 "\033[31m", // ERROR/slow execution
00057                 "\033[32m", // INFO/fast execution
00058                 "\033[33m", // WARNING/medium execution
00059                 "\033[0m\n" // RESET + newline
00060             };
00061
00062             static constexpr std::array<const char *, 2> LOG_LEVEL_STRING = {"INFO", "WARNING"};
00063
00064             Logger() = default;
00065             ~Logger() = default;
00066
00067             [[nodiscard]] static const char *getColorForDuration(const float duration)
00068             {
00069                 return duration < 20.0F
00070                         ? LOG_LEVEL_COLOR[COLOR_INFO]
00071                         : (duration < 90.0F ? LOG_LEVEL_COLOR[COLOR_WARNING] :
    LOG_LEVEL_COLOR[COLOR_ERROR]);
00072             }
00073
00074             [[nodiscard]] static std::string formatLogMessage(LogLevel level, const std::string &message)
00075             {
00076                 const auto inTime = std::chrono::system_clock::to_time_t(std::chrono::system_clock::now());
00077                 std::ostringstream ss;
00078                 ss « "[" « std::put_time(std::localtime(&inTime), "%Y-%m-%d %X") « "] ";
00079                 ss « "[" « LOG_LEVEL_STRING[static_cast<uint8_t>(level)] « "] " « message;
00080                 return ss.str();
```

```
00081             }
00082
00083     }; // class Logger
00084
00085 } // namespace utl
```

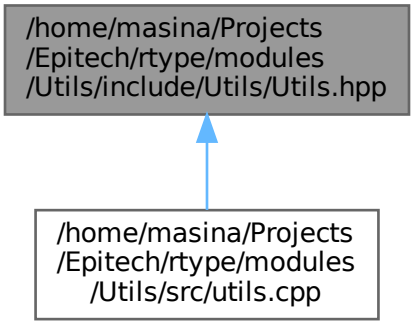
7.33 /home/masina/Projects/Epitech/rtype/modules/Utils/include/Utils/Utils.hpp File Reference

This file contains utility functions.

#include <vector>
Include dependency graph for Utils.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace utl

Functions

- std::vector< char > utl::readFile (const std::string &filename)

7.33.1 Detailed Description

This file contains utility functions.

Definition in file Utils.hpp.

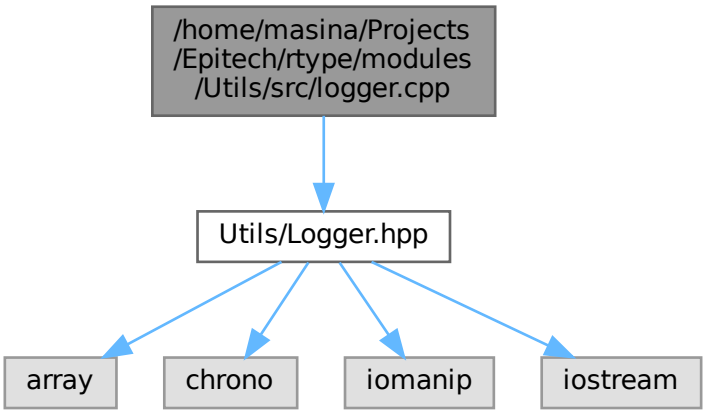
7.34 Utils.hpp

Go to the documentation of this file.

```
00001 ///
00002 /// @file Utils.hpp
00003 /// @brief This file contains utility functions
00004 /// @namespace utl
00005 ///
00006
00007 #pragma once
00008
00009 #include <vector>
00010
00011 namespace utl
00012 {
00013
00014     [[nodiscard]] std::vector<char> readFile(const std::string &filename);
00015
00016 } // namespace utl
```

7.35 /home/masina/Projects/Epitech/rtype/modules/Utils/src/logger.cpp File Reference

#include "Utils/Logger.hpp"
Include dependency graph for logger.cpp:



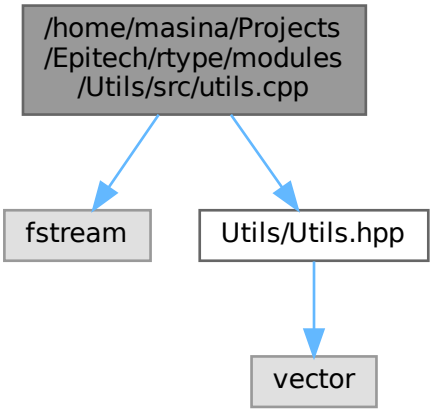
7.36 logger.cpp

Go to the documentation of this file.

```
00001 #ifdef _WIN32
00002 #include <windows.h>
00003 #endif
00004
00005 #include "Utils/Logger.hpp"
00006
00007 void utl::Logger::init()
00008 {
00009 #ifdef _WIN32
00010         const HANDLE hOut = GetStdHandle(STD_OUTPUT_HANDLE);
00011         DWORD dwMode = 0;
00012         if (hOut != INVALID_HANDLE_VALUE && GetConsoleMode(hOut, &dwMode))
00013         {
00014             SetConsoleMode(hOut, dwMode | ENABLE_VIRTUAL_TERMINAL_PROCESSING);
00015         }
00016 #endif
00017 }
```

7.37 /home/masina/Projects/Epitech/rtype/modules/↩ Utils/src/utils.cpp File Reference

#include <fstream>
#include "Utils/Utils.hpp"
Include dependency graph for utils.cpp:



7.38 utils.cpp

Go to the documentation of this file.

```
00001 #include <fstream>
00002
00003 #include "Utils/Utils.hpp"
00004
00005 std::vector<char> utl::readFile(const std::string &filename)
00006 {
00007     std::ifstream file(filename, std::ios::binary | std::ios::ate);
00008     if (!file.is_open())
```

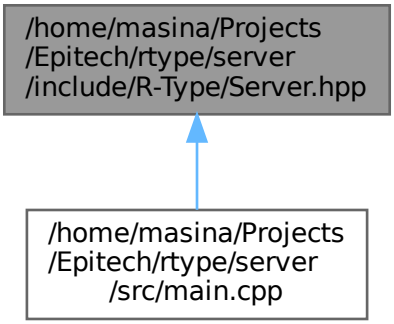
```
00009     {
00010         throw std::runtime_error("failed to open file " + filename);
00011     }
00012     const size_t fileSize = file.tellg();
00013     if (fileSize <= 0)
00014     {
00015         throw std::runtime_error("file " + filename + " is empty");
00016     }
00017     std::vector<char> buffer(fileSize);
00018     file.seekg(0, std::ios::beg);
00019     if (!file.read(buffer.data(), fileSize))
00020     {
00021         throw std::runtime_error("failed to read file " + filename);
00022     }
00023     return buffer;
00024 }
```

7.39 /home/masina/Projects/Epitech/rtype/README.md File Reference

7.40 /home/masina/Projects/Epitech/rtype/server/include/R-Type/Server.hpp File Reference

This file contains the Server class declaration.

This graph shows which files directly or indirectly include this file:



Classes

- class rtp::Server
  Class for the server.

Namespaces

- namespace rtp

7.40.1 Detailed Description

This file contains the Server class declaration.

Definition in file Server.hpp.

7.41 Server.hpp

Go to the documentation of this file.

```
00001 ///
00002 /// @file Server.hpp
00003 /// @brief This file contains the Server class declaration
00004 /// @namespace rtp
00005 ///
00006
00007 #pragma once
00008
00009 namespace rtp
00010 {
00011
00012     ///
00013     /// @class Server
00014     /// @brief Class for the server
00015     /// @namespace rtp
00016     ///
00017     class Server
00018     {
00019
00020         public:
00021             explicit Server(const ArgsConfig &config)
00022             {
00023                 // Initialize server with config
00024                 (void)config;
00025             }
00026             ~Server() = default;
00027
00028             Server(const Server &) = delete;
00029             Server &operator=(const Server &) = delete;
00030             Server(Server &&) = delete;
00031             Server &operator=(Server &&) = delete;
00032
00033         private:
00034     }; // class Server
00035
00036 } // namespace rtp
```

Index