

seastar

0.0.0.1

Generated by Doxygen 1.8.6

Mon Jun 1 2015 16:23:36

Contents

1	Multi-threaded TCP/IP server for simultaneous clients connections	1
1.1	Foreword	1
1.2	Introduction	1
1.3	Support	2
2	Specifications	3
3	Development	5
3.1	Architecture	5
3.2	Coding rules	5
3.3	Naming convention	5
4	Build chain	7
4.1	Requirements	7
4.2	Outputs	7
5	Test and validation	9
5.1	Requirements	9
5.2	Run time	9
6	User manual	11
6.1	Screenshot of built-in help	11
6.2	Screenshot of start using options	12
6.3	Screenshot of connected hosts	13
7	Test list	15
8	Todo List	17
9	Bug List	19
10	Hierarchical Index	21
10.1	Class Hierarchy	21
11	Class Index	23

11.1 Class List	23
12 File Index	25
12.1 File List	25
13 Class Documentation	27
13.1 APPLICATION Class Reference	27
13.1.1 Detailed Description	28
13.1.2 Constructor & Destructor Documentation	28
13.1.2.1 APPLICATION	28
13.1.2.2 ~APPLICATION	28
13.1.3 Member Function Documentation	28
13.1.3.1 Main	28
13.1.3.2 PrintBackTrace	29
13.1.3.3 RunServer	29
13.1.3.4 SetSignalConfig	29
13.1.3.5 SignalHandler	29
13.1.4 Member Data Documentation	29
13.1.4.1 _Running	29
13.1.4.2 Console	29
13.1.4.3 Manager	30
13.1.4.4 Param	30
13.2 CONNECTION Class Reference	30
13.2.1 Detailed Description	31
13.2.2 Constructor & Destructor Documentation	31
13.2.2.1 CONNECTION	31
13.2.2.2 ~CONNECTION	31
13.2.3 Member Function Documentation	31
13.2.3.1 GetHostID	31
13.2.3.2 RunTask	31
13.2.4 Member Data Documentation	32
13.2.4.1 _HostId	32
13.2.4.2 _Manager	32
13.2.4.3 _Socket	32
13.2.4.4 _Thread	32
13.3 CONSOLE Class Reference	32
13.3.1 Detailed Description	34
13.3.2 Constructor & Destructor Documentation	34
13.3.2.1 CONSOLE	34
13.3.2.2 ~CONSOLE	34
13.3.3 Member Function Documentation	34

13.3.3.1 ColBlu	34
13.3.3.2 ColCya	35
13.3.3.3 ColGra	35
13.3.3.4 ColGre	35
13.3.3.5 ColMag	35
13.3.3.6 ColRed	36
13.3.3.7 ColStd	36
13.3.3.8 ColWhi	36
13.3.3.9 ColYel	36
13.3.3.10 GetTimeStamp	37
13.3.3.11 InitLogger	37
13.3.3.12 LogBlank	37
13.3.3.13 LogError	37
13.3.3.14 LogExcept	37
13.3.3.15 LogInfo	37
13.3.3.16 LogSignal	38
13.3.3.17 LogWarn	38
13.3.3.18 PrintHelp	38
13.3.3.19 PrintLogLine	38
13.3.3.20 PrintSplashScreen	39
13.3.3.21 ReleaseLogger	39
13.3.3.22 SetColors	39
13.3.3.23 SetFullPath	39
13.3.3.24 SetVerbose	39
13.3.4 Member Data Documentation	40
13.3.4.1 _Buffer	40
13.3.4.2 _Colors	40
13.3.4.3 _FullPath	40
13.3.4.4 _Lock	40
13.3.4.5 _Stream	40
13.3.4.6 _Verbose	40
13.4 EXCEPTION Class Reference	40
13.4.1 Detailed Description	41
13.4.2 Constructor & Destructor Documentation	41
13.4.2.1 EXCEPTION	41
13.4.2.2 ~EXCEPTION	41
13.4.3 Member Function Documentation	41
13.4.3.1 GetDescription	41
13.4.3.2 what	42
13.4.4 Member Data Documentation	42

13.4.4.1	_Description	42
13.5	MANAGER Class Reference	42
13.5.1	Detailed Description	43
13.5.2	Constructor & Destructor Documentation	43
13.5.2.1	MANAGER	43
13.5.2.2	~MANAGER	43
13.5.3	Member Function Documentation	43
13.5.3.1	Add	43
13.5.3.2	Count	44
13.5.3.3	Create	44
13.5.3.4	Destroy	44
13.5.3.5	NewHostID	45
13.5.3.6	Remove	45
13.5.4	Friends And Related Function Documentation	45
13.5.4.1	APPLICATION	45
13.5.5	Member Data Documentation	45
13.5.5.1	_Container	45
13.5.5.2	_Lock	45
13.6	OBJECT Class Reference	45
13.6.1	Detailed Description	46
13.6.2	Constructor & Destructor Documentation	46
13.6.2.1	OBJECT	46
13.6.2.2	~OBJECT	46
13.6.3	Member Function Documentation	47
13.6.3.1	App	47
13.6.4	Friends And Related Function Documentation	47
13.6.4.1	main	47
13.6.5	Member Data Documentation	48
13.6.5.1	_AppPtr	48
13.6.5.2	_ObjName	48
13.7	PARAMETERS Class Reference	48
13.7.1	Detailed Description	49
13.7.2	Constructor & Destructor Documentation	49
13.7.2.1	PARAMETERS	49
13.7.2.2	~PARAMETERS	49
13.7.3	Member Function Documentation	49
13.7.3.1	GetColors	49
13.7.3.2	GetHelp	49
13.7.3.3	GetServerPort	50
13.7.3.4	GetSplashscreen	50

13.7.3.5	GetVerbose	50
13.7.3.6	Parse	50
13.7.4	Member Data Documentation	51
13.7.4.1	_AlreadyParsed	51
13.7.4.2	_Colors	51
13.7.4.3	_Help	51
13.7.4.4	_PortNum	51
13.7.4.5	_Splashscreen	51
13.7.4.6	_Verbose	51
13.8	SOCKET Class Reference	52
13.8.1	Detailed Description	53
13.8.2	Constructor & Destructor Documentation	53
13.8.2.1	SOCKET	53
13.8.2.2	~SOCKET	53
13.8.2.3	SOCKET	53
13.8.3	Member Function Documentation	53
13.8.3.1	Accept	53
13.8.3.2	Bind	54
13.8.3.3	Connect	55
13.8.3.4	GetId	55
13.8.3.5	GetLocalAddr	55
13.8.3.6	GetRemoteAddr	55
13.8.3.7	IsConnected	56
13.8.3.8	IsDataWaiting	56
13.8.3.9	Listen	56
13.8.3.10	Receive	56
13.8.3.11	Send	56
13.8.3.12	WaitData	57
13.8.4	Member Data Documentation	57
13.8.4.1	_SocketId	57
13.9	THREAD Class Reference	57
13.9.1	Detailed Description	58
13.9.2	Constructor & Destructor Documentation	58
13.9.2.1	THREAD	58
13.9.2.2	~THREAD	58
13.9.3	Member Function Documentation	58
13.9.3.1	Cancel	58
13.9.3.2	Run	59
13.9.3.3	ThreadFunction	59
13.9.4	Member Data Documentation	59

13.9.4.1	<code>_Argument</code>	59
13.9.4.2	<code>_Attr</code>	59
13.9.4.3	<code>_Procedure</code>	59
13.9.4.4	<code>_SigMask</code>	59
13.9.4.5	<code>_ThreadId</code>	60
14	File Documentation	61
14.1	<code>inc/application.h</code> File Reference	61
14.1.1	Detailed Description	61
14.2	<code>inc/connection.h</code> File Reference	61
14.2.1	Detailed Description	62
14.3	<code>inc/console.h</code> File Reference	62
14.3.1	Detailed Description	62
14.3.2	Macro Definition Documentation	62
14.3.2.1	<code>SOURCE_LINE</code>	62
14.3.2.2	<code>SSTR</code>	63
14.3.2.3	<code>STR</code>	63
14.3.3	Enumeration Type Documentation	63
14.3.3.1	<code>LOG_TYPE</code>	63
14.4	<code>inc/exception.h</code> File Reference	63
14.4.1	Detailed Description	63
14.5	<code>inc/manager.h</code> File Reference	64
14.5.1	Detailed Description	64
14.5.2	Typedef Documentation	64
14.5.2.1	<code>CONTAINER</code>	64
14.6	<code>inc/object.h</code> File Reference	64
14.6.1	Detailed Description	65
14.7	<code>inc/parameters.h</code> File Reference	65
14.7.1	Detailed Description	65
14.8	<code>inc/socket.h</code> File Reference	65
14.8.1	Detailed Description	65
14.9	<code>inc/thread.h</code> File Reference	66
14.9.1	Detailed Description	66
14.10	<code>src/application.cpp</code> File Reference	66
14.10.1	Detailed Description	66
14.11	<code>src/connection.cpp</code> File Reference	67
14.11.1	Detailed Description	67
14.11.2	Macro Definition Documentation	67
14.11.2.1	<code>CYCLE_DURATION_MS</code>	67
14.11.2.2	<code>CYCLE_DURATION_US</code>	67

14.11.2.3 SLICE_DURATION_MS	67
14.11.2.4 SLICE_DURATION_US	67
14.12src/console.cpp File Reference	68
14.12.1 Detailed Description	68
14.12.2 Macro Definition Documentation	68
14.12.2.1 CODE_BLK	68
14.12.2.2 CODE_BLU	69
14.12.2.3 CODE_CYA	69
14.12.2.4 CODE_GRA	69
14.12.2.5 CODE_GRE	69
14.12.2.6 CODE_MAG	69
14.12.2.7 CODE_RED	69
14.12.2.8 CODE_WHI	69
14.12.2.9 CODE_YEL	69
14.12.2.10ESC_BLK	69
14.12.2.11ESC_BLU	69
14.12.2.12ESC_CYA	69
14.12.2.13ESC_GRA	70
14.12.2.14ESC_GRE	70
14.12.2.15ESC_MAG	70
14.12.2.16ESC_RED	70
14.12.2.17ESC_SEQ	70
14.12.2.18ESC_STD	70
14.12.2.19ESC_WHI	70
14.12.2.20ESC_YEL	70
14.13src/exception.cpp File Reference	70
14.13.1 Detailed Description	71
14.14src/main.cpp File Reference	71
14.14.1 Detailed Description	71
14.14.2 Function Documentation	71
14.14.2.1 main	71
14.15src/manager.cpp File Reference	72
14.15.1 Detailed Description	72
14.16src/object.cpp File Reference	72
14.16.1 Detailed Description	72
14.17src/parameters.cpp File Reference	73
14.17.1 Detailed Description	73
14.17.2 Macro Definition Documentation	73
14.17.2.1 DEFLT_SERV_PORT	73
14.18src/socket.cpp File Reference	73

14.18.1 Detailed Description	73
14.19src/thread.cpp File Reference	74
14.19.1 Detailed Description	74
Index	75

Chapter 1

Multi-threaded TCP/IP server for simultaneous clients connections

1.1 Foreword

The seastar is an beautiful aquatic animal whom several arms join the center of its body, like several clients can connect to a server. That's why I have chosen this mascot figuring as the logo for my project. Every famous Open Source Project has one so why not mine ?



Figure 1.1: Logo (seastar)

1.2 Introduction

The goal of the project is the developpment of a TCP/IP server which smultaneously deals with several connected clients. Remember that it is only a demonstrative project that is not aimed to be released within a long lifecycle...

1.3 Support

For any question or bug reporting, please contact the author [Olivier de BLIC](#).

Chapter 2

Specifications

Here are the original basic specifications for the project

Write a server in C++ language serving arbitrary number of TCP clients.

- For every TCP client connected a new thread in the server will be spawned. The thread will send a server-wide unique binary 32 bit ID number every second. The ID will be ASCII coded before sending to client and terminated by a new line character. The mechanism for finding unique IDs is performance sensitive and cannot be pre-computed.
- The server also responds new line character received from the client with number of clients connected.
- Ctrl-C will send "Bye\n" to all the connected clients and immediately terminate the server cleanly.
- Limit the usage of global variables.

Chapter 3

Development

Here are the detailed documentation of the software design and the source code.

3.1 Architecture

The program is strongly inspired of a program in Java : everything is an object (no global variable out of any class).

3.2 Coding rules

- No pointer declared without initialization
- Public members of builtin types are forbidden
- No use of default constructors and destructors
- Only one class definition/declaration per file

3.3 Naming convention

- Variables use CamelCase convention.
- Every identifier starting with an underscore stands for a private member.
- File names only use alphanumeric lowercase characters.

Chapter 4

Build chain

Here is the description of the build chain for the outputs (program and documentation)

4.1 Requirements

For the program

All the classic tools for building Linux programs are needed (gcc, g++, ld, make, etc.)

For the documentation

Doxygen and texlive are needed for automatic documentation generation (in **HTML**, **Latex** and **PDF**).

4.2 Outputs

Generated files (objects and binary) are same for release and debug building mode, thus you need to rebuild all the project when changing mode (target **clean** and then **all**).

Use the make command to generate the available targets which are :

- **all** debug or release version (release by default)
- **dep** dependencies generation
- **tarball** backup of the whole project in a tarball
- **clean** cleanup of the project directory

For the debug or release version selection, use the following commands :

- make **clean** && make **all** -DDEBUG
- make **clean** && make **all**

Chapter 5

Test and validation

How to test the program to check the specifications matching.

5.1 Requirements

netcat (or other TCP client) is needed to connect to the server as client from a terminal. Linux OS on a standard PC architecture.

5.2 Run time

Use the dedicated script :

- sh test.sh

Chapter 6

User manual

How to use the program.

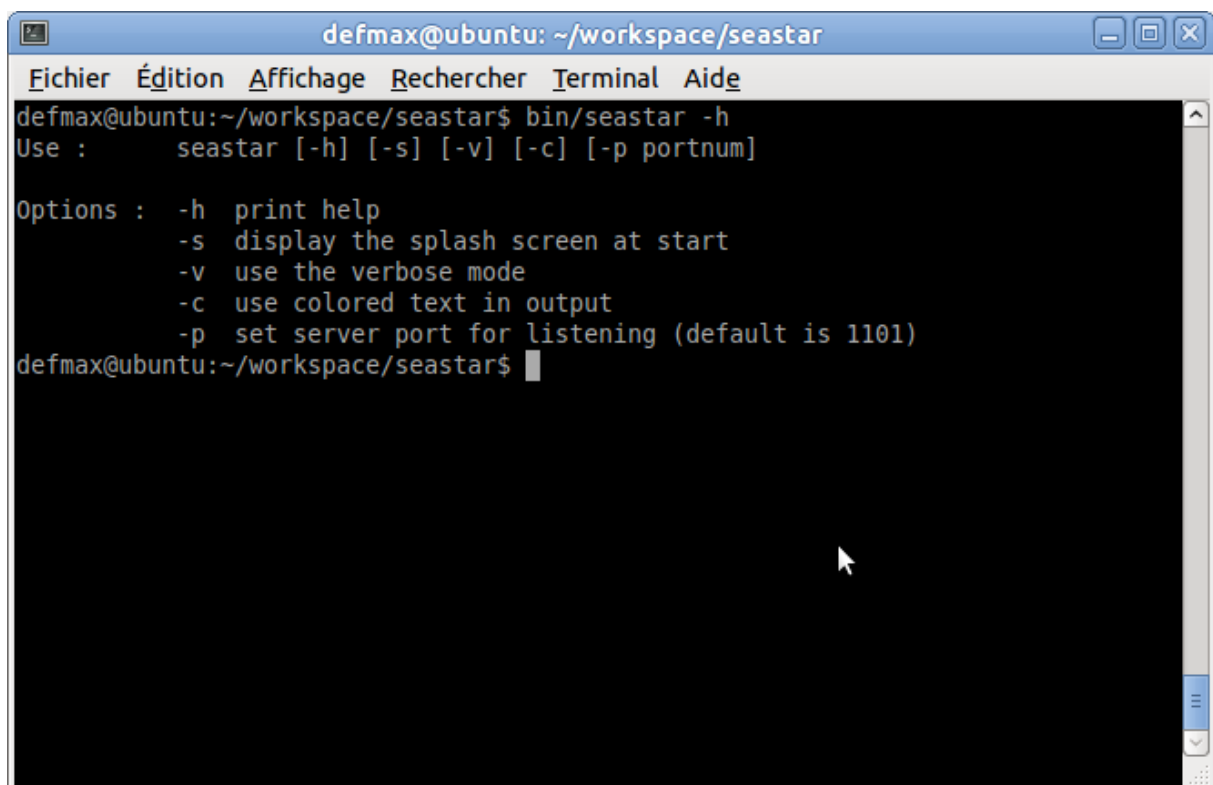
Run **seastar** to wait for incoming connections (options -s -v -c are so nice...) :

- seastar -svc

Use **netcat** to connect to the server as host :

- nc localhost 1101

6.1 Screenshot of built-in help



```
defmax@ubuntu: ~/workspace/seastar
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
defmax@ubuntu:~/workspace/seastar$ bin/seastar -h
Use :      seastar [-h] [-s] [-v] [-c] [-p portnum]

Options : -h  print help
          -s  display the splash screen at start
          -v  use the verbose mode
          -c  use colored text in output
          -p  set server port for listening (default is 1101)
defmax@ubuntu:~/workspace/seastar$
```

Figure 6.1: Help (screenshot)

6.3 Screenshot of connected hosts

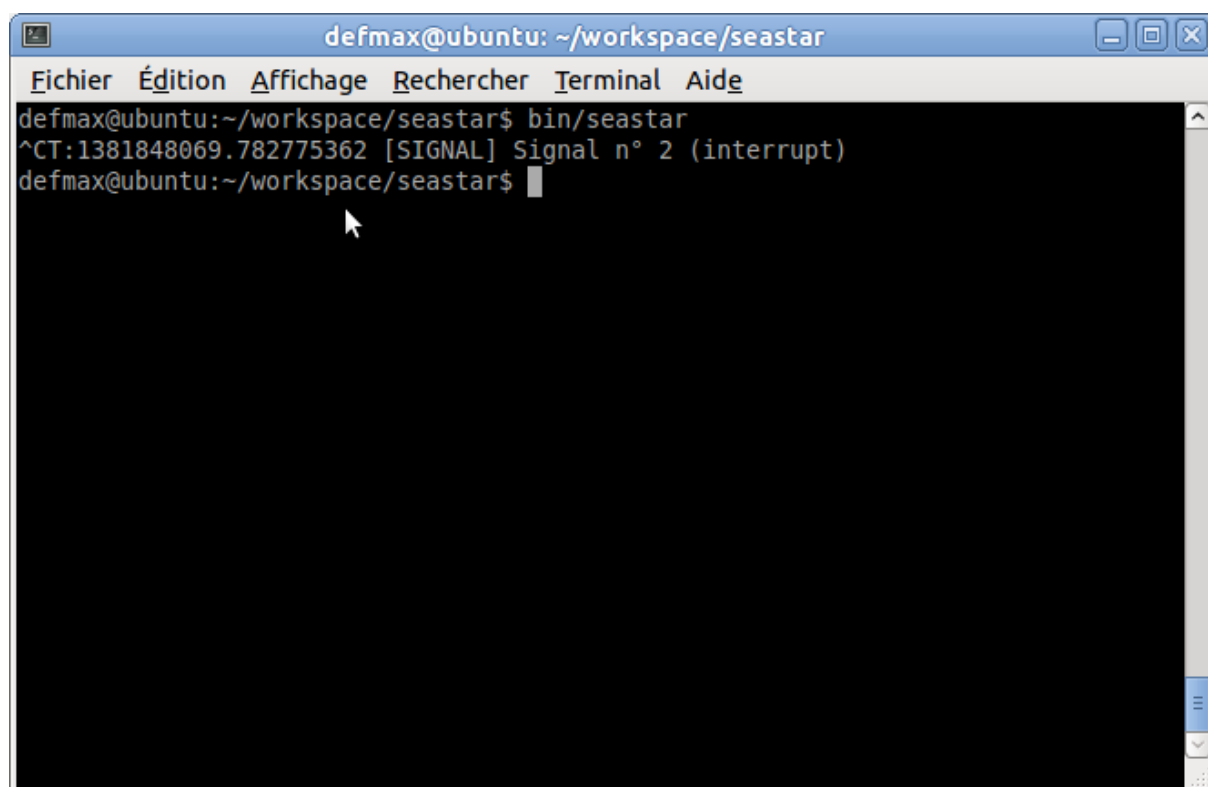
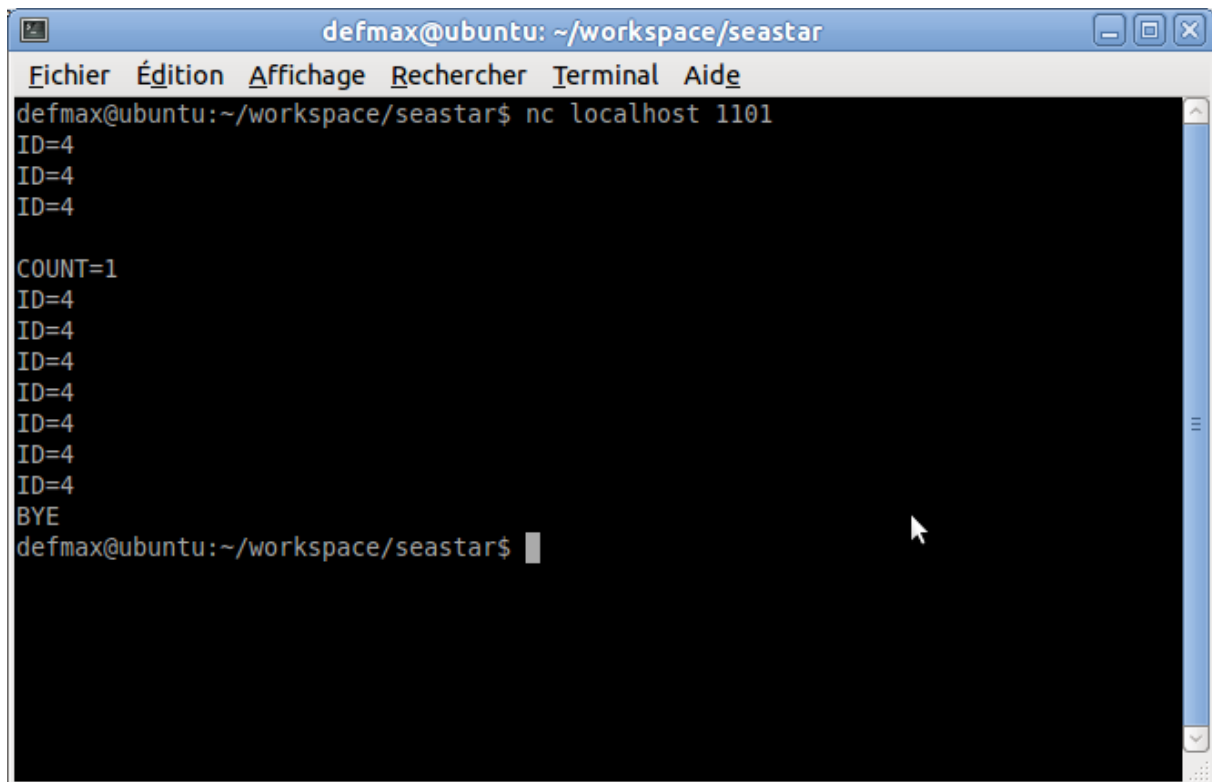


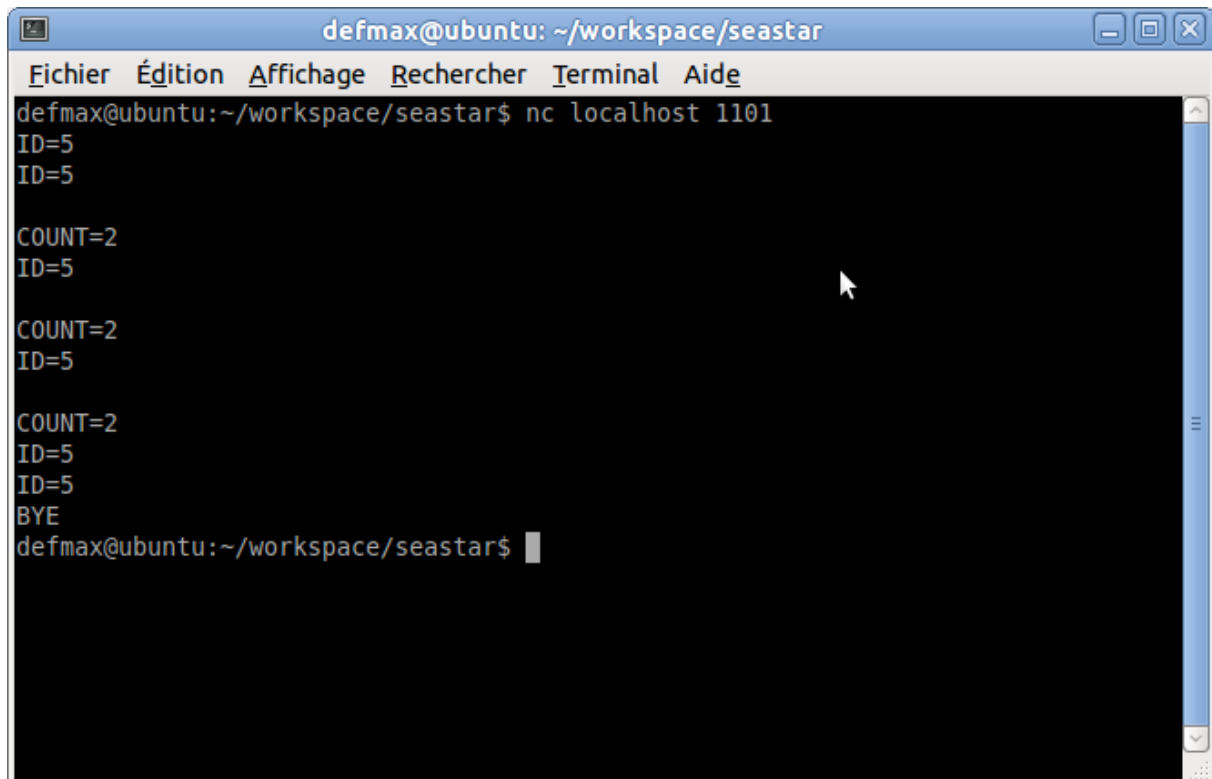
Figure 6.3: Server (screenshot)



```
defmax@ubuntu: ~/workspace/seastar
Fichier Édition Affichage Rechercher Terminal Aide
defmax@ubuntu:~/workspace/seastar$ nc localhost 1101
ID=4
ID=4
ID=4

COUNT=1
ID=4
ID=4
ID=4
ID=4
ID=4
ID=4
ID=4
ID=4
BYE
defmax@ubuntu:~/workspace/seastar$
```

Figure 6.4: Host 1 (screenshot)



```
defmax@ubuntu: ~/workspace/seastar
Fichier Édition Affichage Rechercher Terminal Aide
defmax@ubuntu:~/workspace/seastar$ nc localhost 1101
ID=5
ID=5

COUNT=2
ID=5

COUNT=2
ID=5

COUNT=2
ID=5
ID=5
BYE
defmax@ubuntu:~/workspace/seastar$
```

Figure 6.5: Host 2 (screenshot)

Chapter 7

Test list

Member `main` (int argc, char *argv[])

Client connection

Unique ID

Number of client sent by server

Client disconnection

Chapter 8

Todo List

Member `APPLICATION::Main` (int ArgCnt, char *ArgVal[])

Separate the source code for a generic `APPLICATION` object and for the server

Use the exception handler `catch(std::exception e)` instead of `catch(EXCEPTION e)`

Member `APPLICATION::SetSignalConfig` ()

Improve the signals management

Member `EXCEPTION::EXCEPTION` (const std::string &Description)

Add the source code reference to localize the thrown exception

Member `main` (int argc, char *argv[])

Apply the `const` rules in the whole project

Use the `namespace ss = sea_star;` to declare identifiers dedicated to the project

Member `PARAMETERS::Parse` (int ArgCnt, char *ArgVal[])

Modify the option to disable colors

Member `SOCKET::IsDataWaiting` ()

Should use SIGIO instead of polling ? (Busy waiting against sleeping...)

Member `SOCKET::~~SOCKET` ()

Use the function `shutdown()` instead of `close()`

Chapter 9

Bug List

Member **CONNECTION::RunTask** (void *Arg)

IsDataWaiting() just ensure that reading socket will not block ! Data is not waiting for sure !

Member **SOCKET::IsDataWaiting** ()

poll() will not block as soon as the socket should block !

Chapter 10

Hierarchical Index

10.1 Class Hierarchy

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

CONSOLE	??
exception	
EXCEPTION	??
OBJECT	??
APPLICATION	??
CONNECTION	??
MANAGER	??
SOCKET	??
THREAD	??
PARAMETERS	??

Chapter 11

Class Index

11.1 Class List

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

APPLICATION	Running program and all its components as subobjects	??
CONNECTION	Host connection with socket and thread embedded	??
CONSOLE	Interface for console output and formatted logs	??
EXCEPTION	Generic exception dedicated for the application	??
MANAGER	Connections manager	??
OBJECT	Base class to give objects a name and an access to the object APPLICATION	??
PARAMETERS	Parser to read parameters passed to the main() function	??
SOCKET	Wrapper for sockets use in C++	??
THREAD	Wrapper for POSIX threads use in C++	??

Chapter 12

File Index

12.1 File List

Here is a list of all files with brief descriptions:

inc/ application.h	Header for main application	??
inc/ connection.h	Header for host connection use	??
inc/ console.h	Header for console use	??
inc/ exception.h	Header for exceptions use	??
inc/ manager.h	Header for connections management	??
inc/ object.h	Header for objects using APPLICATION	??
inc/ parameters.h	Header for parameters use	??
inc/ socket.h	Header for sockets use	??
inc/ thread.h	Header for threads use	??
src/ application.cpp	Module for main application	??
src/ connection.cpp	Module for host connection use	??
src/ console.cpp	Module for console use	??
src/ exception.cpp	Module for exceptions use	??
src/ main.cpp	Module containing the main() function	??
src/ manager.cpp	Module for connections management	??
src/ object.cpp	Module for objects using APPLICATION	??
src/ parameters.cpp	Module for parameters use	??
src/ socket.cpp	Module for sockets use	??
src/ thread.cpp	Module for threads use	??

Chapter 13

Class Documentation

13.1 APPLICATION Class Reference

Running program and all its components as subobjects.

```
#include <application.h>
```

Inherits [OBJECT](#).

Public Member Functions

- [APPLICATION](#) ()
Application constructor.
- virtual [~APPLICATION](#) ()
Application destructor.
- int [Main](#) (int ArgCnt, char *ArgVal[])
Entry point of the application.

Public Attributes

- [PARAMETERS](#) Param
Parameters management.
- [CONSOLE](#) Console
Console outputs management.
- [MANAGER](#) Manager
Clients connections management.

Private Member Functions

- void [RunServer](#) (int PortNum)
Server function.
- void [SetSignalConfig](#) ()
Signal configuration setup.

Static Private Member Functions

- static void [SignalHandler](#) (int SigNum)
Signal management function.

- static void [PrintBackTrace](#) ()
Backtrace display.

Private Attributes

- bool [_Running](#)
Flag for state of application.

Additional Inherited Members

13.1.1 Detailed Description

Running program and all its components as subobjects.
Definition at line 27 of file application.h.

13.1.2 Constructor & Destructor Documentation

13.1.2.1 APPLICATION::APPLICATION ()

Application constructor.
Definition at line 31 of file application.cpp.

13.1.2.2 APPLICATION::~~APPLICATION () [virtual]

Application destructor.
Definition at line 41 of file application.cpp.

13.1.3 Member Function Documentation

13.1.3.1 int APPLICATION::Main (int ArgCnt, char * ArgVal[])

Entry point of the application.

Parameters

<i>ArgCnt</i>	Number of arguments
<i>ArgVal</i>	Values of arguments

Returns

EXIT_SUCCESS if in case of success
EXIT_FAILURE if an error occurred

Todo Separate the source code for a generic [APPLICATION](#) object and for the server

Todo Use the exception handler `catch(std::exception e)` instead of `catch(EXCEPTION e)`

Definition at line 56 of file application.cpp.

References [Console](#), [PARAMETERS::GetColors\(\)](#), [PARAMETERS::GetHelp\(\)](#), [PARAMETERS::GetServerPort\(\)](#), [PARAMETERS::GetSplashscreen\(\)](#), [PARAMETERS::GetVerbose\(\)](#), [CONSOLE::LogExcept\(\)](#), [Param](#), [PARAMETERS::Parse\(\)](#), [CONSOLE::PrintHelp\(\)](#), [CONSOLE::PrintSplashScreen\(\)](#), [RunServer\(\)](#), [CONSOLE::SetColors\(\)](#), [SetSignalConfig\(\)](#), and [CONSOLE::SetVerbose\(\)](#).

Referenced by [main\(\)](#).

13.1.3.2 void APPLICATION::PrintBackTrace () [static],[private]

Backtrace display.

Definition at line 210 of file application.cpp.

Referenced by SignalHandler().

13.1.3.3 void APPLICATION::RunServer (int PortNum) [private]

Server function.

Parameters

<i>PortNum</i>	Port number
----------------	-------------

Definition at line 111 of file application.cpp.

References `_Running`, `SOCKET::Accept()`, `SOCKET::Bind()`, `Console`, `MANAGER::Create()`, `SOCKET::Listen()`, `C-ONSOLE::LogInfo()`, and `Manager`.

Referenced by `Main()`.

13.1.3.4 void APPLICATION::SetSignalConfig () [private]

Signal configuration setup.

Todo Improve the signals management

Definition at line 140 of file application.cpp.

References `SignalHandler()`.

Referenced by `Main()`.

13.1.3.5 void APPLICATION::SignalHandler (int SigNum) [static],[private]

Signal management function.

Parameters

<i>SigNum</i>	Signal occurred
---------------	-----------------

Definition at line 183 of file application.cpp.

References `_Running`, `OBJECT::App()`, `Console`, `CONSOLE::LogSignal()`, and `PrintBackTrace()`.

Referenced by `SetSignalConfig()`.

13.1.4 Member Data Documentation

13.1.4.1 bool APPLICATION::_Running [private]

Flag for state of application.

Definition at line 110 of file application.h.

Referenced by `RunServer()`, and `SignalHandler()`.

13.1.4.2 CONSOLE APPLICATION::Console

Console outputs management.

Definition at line 66 of file application.h.

Referenced by SOCKET::Accept(), MANAGER::Add(), SOCKET::Bind(), CONNECTION::CONNECTION(), SOCKET::Listen(), Main(), SOCKET::Receive(), MANAGER::Remove(), RunServer(), CONNECTION::RunTask(), SOCKET::Send(), SignalHandler(), SOCKET::SOCKET(), THREAD::THREAD(), SOCKET::WaitData(), CONNECTION::~~CONNECTION(), and THREAD::~~THREAD().

13.1.4.3 MANAGER APPLICATION::Manager

Clients connections management.

Definition at line 71 of file application.h.

Referenced by RunServer().

13.1.4.4 PARAMETERS APPLICATION::Param

Parameters management.

Definition at line 61 of file application.h.

Referenced by Main().

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

- [inc/application.h](#)
- [src/application.cpp](#)

13.2 CONNECTION Class Reference

Host connection with socket and thread embedded.

```
#include <connection.h>
```

Inherits [OBJECT](#).

Public Member Functions

- [CONNECTION](#) ([MANAGER](#) &Manager, [SOCKET](#) &Socket, int HostID)
Connection constructor.
- virtual [~CONNECTION](#) ()
Connection destructor.
- int [GetHostID](#) () const
Getter for host identifier.

Static Private Member Functions

- static void * [RunTask](#) (void *Arg)
Task for the connection management.

Private Attributes

- const int [_HostId](#)
- [MANAGER](#) & [_Manager](#)
- [SOCKET](#) & [_Socket](#)
- [THREAD](#) & [_Thread](#)

Additional Inherited Members

13.2.1 Detailed Description

Host connection with socket and thread embedded.

Definition at line 32 of file connection.h.

13.2.2 Constructor & Destructor Documentation

13.2.2.1 CONNECTION::CONNECTION (*MANAGER* & *Manager*, *SOCKET* & *Socket*, int *HostID*)

Connection constructor.

Parameters

<i>Manager</i>	Reference to the owner manager
<i>Socket</i>	Reference to the opened socket
<i>HostID</i>	Host identifier

Definition at line 35 of file connection.cpp.

References `OBJECT::_ObjName`, `_Thread`, `OBJECT::App()`, `APPLICATION::Console`, and `THREAD::Run()`.

13.2.2.2 CONNECTION::~CONNECTION () [virtual]

Connection destructor.

Definition at line 50 of file connection.cpp.

References `OBJECT::_ObjName`, `_Socket`, `_Thread`, `OBJECT::App()`, `THREAD::Cancel()`, `APPLICATION::Console`, `SOCKET::IsConnected()`, and `SOCKET::Send()`.

13.2.3 Member Function Documentation

13.2.3.1 int CONNECTION::GetHostID () const

Getter for host identifier.

Returns

Host identifier

Definition at line 70 of file connection.cpp.

References `_HostId`.

Referenced by `MANAGER::Destroy()`.

13.2.3.2 void * CONNECTION::RunTask (void * *Arg*) [static], [private]

Task for the connection management.

Parameters

<i>Arg</i>	Pointer to the concerned connection
------------	-------------------------------------

Returns

Host identifier

Bug IsDataWaiting() just ensure that reading socket will not block ! Data is not waiting for sure !

Definition at line 84 of file connection.cpp.

References _HostId, _Manager, _Socket, OBJECT::App(), APPLICATION::Console, MANAGER::Count(), CYCLE_DURATION_US, MANAGER::Destroy(), SOCKET::GetLocalAddr(), SOCKET::GetRemoteAddr(), SOCKET::IsConnected(), SOCKET::IsDataWaiting(), CONSOLE::LogExcept(), CONSOLE::LogInfo(), SOCKET::Receive(), SOCKET::Send(), SLICE_DURATION_US, and SOURCE_LINE.

13.2.4 Member Data Documentation**13.2.4.1** `const int CONNECTION::_HostId` `[private]`

Definition at line 78 of file connection.h.

Referenced by GetHostID(), and RunTask().

13.2.4.2 `MANAGER& CONNECTION::_Manager` `[private]`

Definition at line 80 of file connection.h.

Referenced by RunTask().

13.2.4.3 `SOCKET& CONNECTION::_Socket` `[private]`

Definition at line 81 of file connection.h.

Referenced by RunTask(), and ~CONNECTION().

13.2.4.4 `THREAD& CONNECTION::_Thread` `[private]`

Definition at line 82 of file connection.h.

Referenced by CONNECTION(), and ~CONNECTION().

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

- [inc/connection.h](#)
- [src/connection.cpp](#)

13.3 CONSOLE Class Reference

Interface for console output and formatted logs.

```
#include <console.h>
```

Public Member Functions

- [CONSOLE](#) ()
Console constructor.
- [~CONSOLE](#) ()
Console destructor.

- void [SetColor](#)s (bool Enabled)
Configuration setter for colored mode.
- void [SetVerbose](#) (bool Enabled)
Configuration setter for verbose mode.
- void [SetFullPath](#) (bool Enabled)
Configuration setter for fullpath display mode.
- void [LogBlank](#) ()
Blank log.
- void [LogInfo](#) (std::string Msg, std::string Source="")
Information log.
- void [LogWarn](#) (std::string Msg, std::string Source="")
Warning log.
- void [LogError](#) (std::string Msg, std::string Source="")
Error log.
- void [LogExcept](#) (EXCEPTION Exception, std::string Source="")
Exception occurrence log.
- void [LogSignal](#) (int SigNum)
Signal occurrence log.
- void [PrintSplashScreen](#) ()
Print the splashscreen at start.
- void [PrintHelp](#) ()
Print the help.
- void [PrintLogLine](#) (LOG_TYPE Type, std::string LogData="", std::string Source="")
Signal occurrence log.

Private Member Functions

- void [InitLogger](#) ()
Begin of a new log operation.
- void [ReleaseLogger](#) ()
End of a new log operation.
- std::string [GetTimeStamp](#) ()
Get the current timestamp.
- const char * [ColStd](#) ()
Terminal on the fly configuration (standard display)
- const char * [ColGra](#) ()
Terminal on the fly configuration (gray display)
- const char * [ColRed](#) ()
Terminal on the fly configuration (red display)
- const char * [ColGre](#) ()
Terminal on the fly configuration (green display)
- const char * [ColYel](#) ()
Terminal on the fly configuration (yellow display)
- const char * [ColBlu](#) ()
Terminal on the fly configuration (blue display)
- const char * [ColMag](#) ()
Terminal on the fly configuration (magenta display)
- const char * [ColCya](#) ()
Terminal on the fly configuration (cyan display)
- const char * [ColWhi](#) ()
Terminal on the fly configuration (white display)

Private Attributes

- pthread_mutex_t [_Lock](#)
Mutex to prevent of lines cutting due to multi-threading.
- bool [_Colors](#)
Flag for color mode.
- bool [_Verbose](#)
Flag for verbose mode.
- bool [_FullPath](#)
Flag for fullpath mode.
- std::ostringstream [_Buffer](#)
Buffer for log line construction.
- std::ostream & [_Stream](#)
Stream for log line printing.

13.3.1 Detailed Description

Interface for console output and formatted logs.

Definition at line 48 of file console.h.

13.3.2 Constructor & Destructor Documentation

13.3.2.1 `CONSOLE::CONSOLE ()`

Console constructor.

Definition at line 53 of file console.cpp.

References [_Lock](#).

13.3.2.2 `CONSOLE::~~CONSOLE ()`

Console destructor.

Definition at line 64 of file console.cpp.

References [_Lock](#).

13.3.3 Member Function Documentation

13.3.3.1 `const char * CONSOLE::ColBlu ()` [private]

Terminal on the fly configuration (blue display)

Returns

ANSI escape sequence if colored mode enabled
The "" string if colored mode disabled

Definition at line 565 of file console.cpp.

References [_Colors](#), and [ESC_BLU](#).

Referenced by [PrintLogLine\(\)](#).

13.3.3.2 `const char * CONSOLE::ColCya () [private]`

Terminal on the fly configuration (cyan display)

Returns

ANSI escape sequence if colored mode enabled
The "" string if colored mode disabled

Definition at line 585 of file console.cpp.

References `_Colors`, and `ESC_CYA`.

Referenced by `PrintLogLine()`, and `PrintSplashScreen()`.

13.3.3.3 `const char * CONSOLE::ColGra () [private]`

Terminal on the fly configuration (gray display)

Returns

ANSI escape sequence if colored mode enabled
The "" string if colored mode disabled

Definition at line 525 of file console.cpp.

References `_Colors`, and `ESC_GRA`.

Referenced by `PrintLogLine()`.

13.3.3.4 `const char * CONSOLE::ColGre () [private]`

Terminal on the fly configuration (green display)

Returns

ANSI escape sequence if colored mode enabled
The "" string if colored mode disabled

Definition at line 545 of file console.cpp.

References `_Colors`, and `ESC_GRE`.

Referenced by `PrintLogLine()`.

13.3.3.5 `const char * CONSOLE::ColMag () [private]`

Terminal on the fly configuration (magenta display)

Returns

ANSI escape sequence if colored mode enabled
The "" string if colored mode disabled

Definition at line 575 of file console.cpp.

References `_Colors`, and `ESC_MAG`.

Referenced by `PrintLogLine()`.

13.3.3.6 `const char * CONSOLE::ColRed () [private]`

Terminal on the fly configuration (red display)

Returns

ANSI escape sequence if colored mode enabled
The "" string if colored mode disabled

Definition at line 535 of file console.cpp.

References `_Colors`, and `ESC_RED`.

Referenced by `PrintLogLine()`, and `PrintSplashScreen()`.

13.3.3.7 `const char * CONSOLE::ColStd () [private]`

Terminal on the fly configuration (standard display)

Returns

ANSI escape sequence if colored mode enabled
The "" string if colored mode disabled

Definition at line 515 of file console.cpp.

References `_Colors`, and `ESC_STD`.

Referenced by `PrintLogLine()`, and `PrintSplashScreen()`.

13.3.3.8 `const char * CONSOLE::ColWhi () [private]`

Terminal on the fly configuration (white display)

Returns

ANSI escape sequence if colored mode enabled
The "" string if colored mode disabled

Definition at line 595 of file console.cpp.

References `_Colors`, and `ESC_WHI`.

Referenced by `PrintLogLine()`.

13.3.3.9 `const char * CONSOLE::ColYel () [private]`

Terminal on the fly configuration (yellow display)

Returns

ANSI escape sequence if colored mode enabled
The "" string if colored mode disabled

Definition at line 555 of file console.cpp.

References `_Colors`, and `ESC_YEL`.

Referenced by `PrintLogLine()`, and `PrintSplashScreen()`.

13.3.3.10 `std::string CONSOLE::GetTimeStamp () [private]`

Get the current timestamp.

Returns

Current timestamp

Definition at line 490 of file console.cpp.

Referenced by PrintLogLine().

13.3.3.11 `void CONSOLE::InitLogger () [private]`

Begin of a new log operation.

Definition at line 447 of file console.cpp.

References `_Buffer`, and `_Lock`.

Referenced by PrintHelp(), PrintLogLine(), and PrintSplashScreen().

13.3.3.12 `void CONSOLE::LogBlank ()`

Blank log.

Definition at line 191 of file console.cpp.

References `LOG_BLANK`, and PrintLogLine().

13.3.3.13 `void CONSOLE::LogError (std::string Msg, std::string Source = " ")`

Error log.

Parameters

<i>Msg</i>	Error message
<i>Source</i>	Source line from where the function is call

Definition at line 233 of file console.cpp.

References `LOG_ERROR`, and PrintLogLine().

13.3.3.14 `void CONSOLE::LogExcept (EXCEPTION Exception, std::string Source = " ")`

Exception occurrence log.

Parameters

<i>Exception</i>	Exception object
<i>Source</i>	Source line from where the function is call

Definition at line 246 of file console.cpp.

References `EXCEPTION::GetDescription()`, `LOG_EXCEPT`, and PrintLogLine().

Referenced by `APPLICATION::Main()`, and `CONNECTION::RunTask()`.

13.3.3.15 `void CONSOLE::LogInfo (std::string Msg, std::string Source = " ")`

Information log.

Parameters

<i>Msg</i>	Information message
<i>Source</i>	Source line from where the function is call

Definition at line 204 of file console.cpp.

References `_Verbose`, `LOG_INFO`, and `PrintLogLine()`.

Referenced by `SOCKET::Accept()`, `MANAGER::Add()`, `SOCKET::Bind()`, `SOCKET::Listen()`, `SOCKET::Receive()`, `MANAGER::Remove()`, `APPLICATION::RunServer()`, `CONNECTION::RunTask()`, `SOCKET::Send()`, `SOCKET::SOCKET()`, and `SOCKET::WaitData()`.

13.3.3.16 void `CONSOLE::LogSignal (int SigNum)`

Signal occurrence log.

Parameters

<i>SigNum</i>	Signal number
---------------	---------------

Definition at line 258 of file console.cpp.

References `LOG_SIGNAL`, and `PrintLogLine()`.

Referenced by `APPLICATION::SignalHandler()`.

13.3.3.17 void `CONSOLE::LogWarn (std::string Msg, std::string Source = " ")`

Warning log.

Parameters

<i>Msg</i>	Warning message
<i>Source</i>	Source line from where the function is call

Definition at line 220 of file console.cpp.

References `LOG_WARN`, and `PrintLogLine()`.

13.3.3.18 void `CONSOLE::PrintHelp ()`

Print the help.

Definition at line 345 of file console.cpp.

References `_Buffer`, `InitLogger()`, and `ReleaseLogger()`.

Referenced by `APPLICATION::Main()`.

13.3.3.19 void `CONSOLE::PrintLogLine (LOG_TYPE Type, std::string LogData = " ", std::string Source = " ")`

Signal occurrence log.

Parameters

<i>Type</i>	Type of log
<i>LogData</i>	Undefined
<i>Source</i>	Source line from where the function is call

Definition at line 371 of file console.cpp.

References `_Buffer`, `_FullPath`, `ColBlu()`, `ColCya()`, `ColGra()`, `ColGre()`, `ColMag()`, `ColRed()`, `ColStd()`, `ColWhi()`, `ColYel()`, `GetTimeStamp()`, `InitLogger()`, `LOG_BLANK`, `LOG_CTOR`, `LOG_DEBUG`, `LOG_DTOR`, `LOG_ERROR`,

LOG_EXCEPT, LOG_INFO, LOG_SIGNAL, LOG_WARN, and ReleaseLogger().

Referenced by LogBlank(), LogError(), LogExcept(), LogInfo(), LogSignal(), and LogWarn().

13.3.3.20 void CONSOLE::PrintSplashScreen ()

Print the splashscreen at start.

Definition at line 294 of file console.cpp.

References _Buffer, ColCya(), ColRed(), ColStd(), ColYel(), InitLogger(), and ReleaseLogger().

Referenced by APPLICATION::Main().

13.3.3.21 void CONSOLE::ReleaseLogger () [private]

End of a new log operation.

Definition at line 460 of file console.cpp.

References _Buffer, _Lock, and _Stream.

Referenced by PrintHelp(), PrintLogLine(), and PrintSplashScreen().

13.3.3.22 void CONSOLE::SetColors (bool *Enabled*)

Configuration setter for colored mode.

Parameters

<i>Enabled</i>	true for enable, false for disable
----------------	------------------------------------

Definition at line 76 of file console.cpp.

References _Colors.

Referenced by APPLICATION::Main().

13.3.3.23 void CONSOLE::SetFullPath (bool *Enabled*)

Configuration setter for fullpath display mode.

Parameters

<i>Enabled</i>	true for enable, false for disable
----------------	------------------------------------

Definition at line 100 of file console.cpp.

References _FullPath.

13.3.3.24 void CONSOLE::SetVerbose (bool *Enabled*)

Configuration setter for verbose mode.

Parameters

<i>Enabled</i>	true for enable, false for disable
----------------	------------------------------------

Definition at line 88 of file console.cpp.

References _Verbose.

Referenced by APPLICATION::Main().

13.3.4 Member Data Documentation

13.3.4.1 `std::ostringstream` `CONSOLE::_Buffer` `[private]`

Buffer for log line construction.

Definition at line 372 of file `console.h`.

Referenced by `InitLogger()`, `PrintHelp()`, `PrintLogLine()`, `PrintSplashScreen()`, and `ReleaseLogger()`.

13.3.4.2 `bool` `CONSOLE::_Colors` `[private]`

Flag for color mode.

Definition at line 363 of file `console.h`.

Referenced by `ColBlu()`, `ColCya()`, `ColGra()`, `ColGre()`, `ColMag()`, `ColRed()`, `ColStd()`, `ColWhi()`, `ColYel()`, and `SetColors()`.

13.3.4.3 `bool` `CONSOLE::_FullPath` `[private]`

Flag for fullpath mode.

Definition at line 369 of file `console.h`.

Referenced by `PrintLogLine()`, and `SetFullPath()`.

13.3.4.4 `pthread_mutex_t` `CONSOLE::_Lock` `[private]`

Mutex to prevent of lines cutting due to multi-threading.

Definition at line 360 of file `console.h`.

Referenced by `CONSOLE()`, `InitLogger()`, `ReleaseLogger()`, and `~CONSOLE()`.

13.3.4.5 `std::ostream&` `CONSOLE::_Stream` `[private]`

Stream for log line printing.

Definition at line 375 of file `console.h`.

Referenced by `ReleaseLogger()`.

13.3.4.6 `bool` `CONSOLE::_Verbose` `[private]`

Flag for verbose mode.

Definition at line 366 of file `console.h`.

Referenced by `LogInfo()`, and `SetVerbose()`.

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

- `inc/console.h`
- `src/console.cpp`

13.4 EXCEPTION Class Reference

Generic exception dedicated for the application.

```
#include <exception.h>
```

Inherits exception.

Public Member Functions

- [EXCEPTION](#) (const std::string &Description)
Exception constructor.
- virtual [~EXCEPTION](#) () throw ()
Exception destructor.
- const std::string & [GetDescription](#) () const throw ()
Getter for exception description (customized)
- virtual const char * [what](#) () const throw ()
Getter for exception description (standard)

Private Attributes

- std::string [_Description](#)
Exception description.

13.4.1 Detailed Description

Generic exception dedicated for the application.

Definition at line 25 of file exception.h.

13.4.2 Constructor & Destructor Documentation

13.4.2.1 EXCEPTION::EXCEPTION (const std::string & *Description*) [explicit]

Exception constructor.

Parameters

<i>Description</i>	Short description of the cause which has thrown the exception
--------------------	---

Todo Add the source code reference to localize the thrown exception

Definition at line 26 of file exception.cpp.

13.4.2.2 EXCEPTION::~~EXCEPTION () throw) [virtual]

Exception destructor.

Definition at line 37 of file exception.cpp.

13.4.3 Member Function Documentation

13.4.3.1 const std::string & EXCEPTION::GetDescription () const throw)

Getter for exception description (customized)

Returns

Description of the exception (C++ string)

Definition at line 48 of file exception.cpp.

References `_Description`.

Referenced by `CONSOLE::LogExcept()`.

13.4.3.2 `const char * EXCEPTION::what () const throw)` `[virtual]`

Getter for exception description (standard)

Returns

Description of the exception (ANSI C string)

Definition at line 60 of file exception.cpp.

References `_Description`.

13.4.4 Member Data Documentation

13.4.4.1 `std::string EXCEPTION::_Description` `[private]`

Exception description.

Definition at line 66 of file exception.h.

Referenced by `GetDescription()`, and `what()`.

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

- [inc/exception.h](#)
- [src/exception.cpp](#)

13.5 MANAGER Class Reference

Connections manager.

```
#include <manager.h>
```

Inherits [OBJECT](#).

Public Member Functions

- `int Create (SOCKET &)`
Creation of new connection.
- `void Destroy (int HostId)`
Destruction of a connection.
- `int Count () const`
Getter for the current number of connections.

Private Member Functions

- [MANAGER](#) ()
Manager constructor.
- virtual [~MANAGER](#) ()
Manager destructor.
- int [NewHostID](#) ()
Generator for new host identifier.
- void [Add](#) ([CONNECTION](#) *Connection)
Add a connection to the container.
- void [Remove](#) ([CONNECTION](#) *Connection)
Remove a connection from the container.

Private Attributes

- pthread_mutex_t [_Lock](#)
- [CONTAINER](#) _Container

Friends

- class [APPLICATION](#)

Additional Inherited Members

13.5.1 Detailed Description

Connections manager.

Definition at line 37 of file manager.h.

13.5.2 Constructor & Destructor Documentation

13.5.2.1 `MANAGER::MANAGER()` `[private]`

Manager constructor.

Definition at line 29 of file manager.cpp.

References `_Lock`.

13.5.2.2 `MANAGER::~~MANAGER()` `[private],[virtual]`

Manager destructor.

Definition at line 40 of file manager.cpp.

References `_Container`, and `_Lock`.

13.5.3 Member Function Documentation

13.5.3.1 `void MANAGER::Add(CONNECTION * Connection)` `[private]`

Add a connection to the container.

Parameters

<i>Connection</i>	Pointer to the connection
-------------------	---------------------------

Definition at line 110 of file manager.cpp.

References `_Container`, `_Lock`, `OBJECT::App()`, `APPLICATION::Console`, `CONSOLE::LogInfo()`, and `SOURCE_LINE`.

Referenced by `Create()`.

13.5.3.2 int MANAGER::Count () const

Getter for the current number of connections.

Returns

Number of connections in the container

Definition at line 164 of file manager.cpp.

References `_Container`.

Referenced by `CONNECTION::RunTask()`.

13.5.3.3 int MANAGER::Create (SOCKET & Socket)

Creation of new connection.

Parameters

<i>Socket</i>	Socket already opened
---------------	-----------------------

Returns

Host identifier

Parameters

<i>Socket</i>	Socket already opened
---------------	-----------------------

Returns

Host identifier of newly created connection

Definition at line 62 of file manager.cpp.

References `Add()`, and `SOCKET::GetId()`.

Referenced by `APPLICATION::RunServer()`.

13.5.3.4 void MANAGER::Destroy (int HostId)

Destruction of a connection.

Parameters

<i>HostId</i>	Host identifier of connection to destroy
---------------	--

Definition at line 84 of file manager.cpp.

References `_Container`, `CONNECTION::GetHostID()`, and `Remove()`.

Referenced by `CONNECTION::RunTask()`.

13.5.3.5 int MANAGER::NewHostID () [private]

Generator for new host identifier.

Returns

New host identifier

Definition at line 176 of file manager.cpp.

13.5.3.6 void MANAGER::Remove (CONNECTION * Connection) [private]

Remove a connection from the container.

Parameters

<i>Connection</i>	Pointer to the connection
-------------------	---------------------------

Definition at line 135 of file manager.cpp.

References `_Container`, `_Lock`, `OBJECT::App()`, `APPLICATION::Console`, `CONSOLE::LogInfo()`, and `SOURCE_LINE`.

Referenced by `Destroy()`.

13.5.4 Friends And Related Function Documentation

13.5.4.1 friend class APPLICATION [friend]

Definition at line 39 of file manager.h.

13.5.5 Member Data Documentation

13.5.5.1 CONTAINER MANAGER::_Container [private]

Definition at line 120 of file manager.h.

Referenced by `Add()`, `Count()`, `Destroy()`, `Remove()`, and `~MANAGER()`.

13.5.5.2 pthread_mutex_t MANAGER::_Lock [private]

Definition at line 118 of file manager.h.

Referenced by `Add()`, `MANAGER()`, `Remove()`, and `~MANAGER()`.

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

- [inc/manager.h](#)
- [src/manager.cpp](#)

13.6 OBJECT Class Reference

Base class to give objects a name and an access to the object [APPLICATION](#).

```
#include <object.h>
```

Inherited by [APPLICATION](#), [CONNECTION](#), [MANAGER](#), [SOCKET](#), and [THREAD](#).

Protected Member Functions

- [OBJECT](#) (std::string MyName="<unnamed>")
Object constructor.
- virtual [~OBJECT](#) ()
Object destructor.

Static Protected Member Functions

- static [APPLICATION](#) & [App](#) ()
Getter for owning application.

Protected Attributes

- const std::string [_ObjName](#)
Name of current object (for logs and debug)

Static Private Attributes

- static [APPLICATION](#) * [_AppPtr](#) = NULL
Pointer to owning application object (unique for all objects)

Friends

- int [main](#) (int argc, char *argv[])
Entry point of the program.

13.6.1 Detailed Description

Base class to give objects a name and an access to the object [APPLICATION](#).

Definition at line 31 of file object.h.

13.6.2 Constructor & Destructor Documentation

13.6.2.1 [OBJECT::OBJECT](#) (std::string *MyName* = "<unnamed>") [protected]

Object constructor.

Parameters

<i>MyName</i>	Name of the object
---------------	--------------------

Definition at line 26 of file object.cpp.

13.6.2.2 [OBJECT::~~OBJECT](#) () [protected],[virtual]

Object destructor.

Definition at line 36 of file object.cpp.

13.6.3 Member Function Documentation

13.6.3.1 APPLICATION & OBJECT::App () [static],[protected]

Getter for owning application.

Returns

Reference to application

Note

This is the solution chosen to give to any object a way to access the master [APPLICATION](#) object which owns it.

Definition at line 49 of file object.cpp.

References `_AppPtr`.

Referenced by `SOCKET::Accept()`, `MANAGER::Add()`, `SOCKET::Bind()`, `CONNECTION::CONNECTION()`, `SOCKET::Listen()`, `SOCKET::Receive()`, `MANAGER::Remove()`, `CONNECTION::RunTask()`, `SOCKET::Send()`, `APPLICATION::SignalHandler()`, `SOCKET::SOCKET()`, `THREAD::THREAD()`, `SOCKET::WaitData()`, `CONNECTION::~CONNECTION()`, and `THREAD::~THREAD()`.

13.6.4 Friends And Related Function Documentation

13.6.4.1 int main (int argc, char * argv[]) [friend]

Entry point of the program.

Todo Apply the **const** rules in the whole project

Use the **namespace** `ss = sea_star`; to declare identifiers dedicated to the project

heading1 Client connection

Unique ID

Number of client sent by server

Client disconnection

Parameters

<i>argc</i>	Number of arguments
<i>argv</i>	Values of arguments

Returns

`EXIT_SUCCESS` if in case of success

`EXIT_FAILURE` if an error occurred

Remarks

This function is just an interface between the caller and the objet [APPLICATION](#) which is the core of the program.

Definition at line 44 of file main.cpp.

13.6.5 Member Data Documentation

13.6.5.1 APPLICATION * OBJECT::_AppPtr = NULL [static], [private]

Pointer to owning application object (unique for all objects)

Definition at line 74 of file object.h.

Referenced by App(), and main().

13.6.5.2 const std::string OBJECT::_ObjName [protected]

Name of current object (for logs and debug)

Definition at line 67 of file object.h.

Referenced by CONNECTION::CONNECTION(), SOCKET::SOCKET(), THREAD::THREAD(), CONNECTION::~~CONNECTION(), and THREAD::~~THREAD().

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

- [inc/object.h](#)
- [src/object.cpp](#)

13.7 PARAMETERS Class Reference

Parser to read parameters passed to the [main\(\)](#) function.

```
#include <parameters.h>
```

Public Member Functions

- [PARAMETERS](#) ()
Parameters constructor.
- [~PARAMETERS](#) ()
Parameters destructor.
- void [Parse](#) (int ArgCnt, char *ArgVal[])
Parser for arguments.
- bool [GetSplashscreen](#) () const
Getter for splashscreen display.
- bool [GetColors](#) () const
Getter for colors use.
- bool [GetHelp](#) () const
Getter for help query.
- bool [GetVerbose](#) () const
Getter for verbose mode.
- unsigned short [GetServerPort](#) () const
Getter for port number.

Private Attributes

- bool [_AlreadyParsed](#)
- bool [_Splashscreen](#)
- bool [_Colors](#)

- [bool _Help](#)
- [bool _Verbose](#)
- [unsigned short _PortNum](#)

13.7.1 Detailed Description

Parser to read parameters passed to the [main\(\)](#) function.

Definition at line 24 of file parameters.h.

13.7.2 Constructor & Destructor Documentation

13.7.2.1 PARAMETERS::PARAMETERS ()

Parameters constructor.

Definition at line 29 of file parameters.cpp.

13.7.2.2 PARAMETERS::~~PARAMETERS ()

Parameters destructor.

Definition at line 39 of file parameters.cpp.

13.7.3 Member Function Documentation

13.7.3.1 bool PARAMETERS::GetColors () const

Getter for colors use.

Returns

true if enabled
false if disabled

Definition at line 148 of file parameters.cpp.

References [_Colors](#).

Referenced by [APPLICATION::Main\(\)](#).

13.7.3.2 bool PARAMETERS::GetHelp () const

Getter for help query.

Returns

true if enabled
false if disabled

Definition at line 161 of file parameters.cpp.

References [_Help](#).

Referenced by [APPLICATION::Main\(\)](#).

13.7.3.3 unsigned short PARAMETERS::GetServerPort () const

Getter for port number.

Returns

Port number

Definition at line 186 of file parameters.cpp.

References `_PortNum`.

Referenced by `APPLICATION::Main()`.

13.7.3.4 bool PARAMETERS::GetSplashscreen () const

Getter for splashscreen display.

Returns

true if enabled
false if disabled

Definition at line 135 of file parameters.cpp.

References `_Splashscreen`.

Referenced by `APPLICATION::Main()`.

13.7.3.5 bool PARAMETERS::GetVerbose () const

Getter for verbose mode.

Returns

true if enabled
false if disabled

Definition at line 174 of file parameters.cpp.

References `_Verbose`.

Referenced by `APPLICATION::Main()`.

13.7.3.6 void PARAMETERS::Parse (int ArgCnt, char * ArgVal[])

Parser for arguments.

Parameters

<i>ArgCnt</i>	Number of arguments
<i>ArgVal</i>	Values of arguments

Precondition

The arguments should be ones passed to the [main\(\)](#) function.

Postcondition

The options of the program are defined according to the passed arguments.

Todo Modify the option to disable colors

Definition at line 55 of file parameters.cpp.

References `_AlreadyParsed`, `_Colors`, `_Help`, `_PortNum`, `_Splashscreen`, and `_Verbose`.

Referenced by `APPLICATION::Main()`.

13.7.4 Member Data Documentation

13.7.4.1 `bool PARAMETERS::_AlreadyParsed` `[private]`

Definition at line 107 of file parameters.h.

Referenced by `Parse()`.

13.7.4.2 `bool PARAMETERS::_Colors` `[private]`

Definition at line 111 of file parameters.h.

Referenced by `GetColors()`, and `Parse()`.

13.7.4.3 `bool PARAMETERS::_Help` `[private]`

Definition at line 113 of file parameters.h.

Referenced by `GetHelp()`, and `Parse()`.

13.7.4.4 `unsigned short PARAMETERS::_PortNum` `[private]`

Definition at line 117 of file parameters.h.

Referenced by `GetServerPort()`, and `Parse()`.

13.7.4.5 `bool PARAMETERS::_Splashscreen` `[private]`

Definition at line 109 of file parameters.h.

Referenced by `GetSplashscreen()`, and `Parse()`.

13.7.4.6 `bool PARAMETERS::_Verbose` `[private]`

Definition at line 115 of file parameters.h.

Referenced by `GetVerbose()`, and `Parse()`.

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

- [inc/parameters.h](#)
- [src/parameters.cpp](#)

13.8 SOCKET Class Reference

Wrapper for sockets use in C++.

```
#include <socket.h>
```

Inherits [OBJECT](#).

Public Member Functions

- [SOCKET](#) ()
Socket constructor.
- virtual [~SOCKET](#) ()
Socket destructor.
- std::string [GetLocalAddr](#) () const
Local IP address getter.
- std::string [GetRemoteAddr](#) () const
Remote IP address getter.
- void [Bind](#) (unsigned short PortNum)
Binds the socket to local address (server mode)
- void [Listen](#) ()
Listens the socket (server mode)
- void [Connect](#) (std::string IpAddr, unsigned short PortNum)
Connect the socket to remote address (client mode)
- [SOCKET](#) & [Accept](#) ()
Accepts a connection (server mode)
- bool [IsConnected](#) ()
Tells if the socket is connected.
- bool [IsDataWaiting](#) ()
Tells if data is waiting in the buffer.
- bool [WaitData](#) ()
- void [Send](#) (const std::string &Data)
Sends data to socket.
- std::string [Receive](#) ()
Sends data to socket.
- int [GetId](#) () const
Socket identifier getter.

Private Member Functions

- [SOCKET](#) (int SocketId)
Socket constructor (reserved for accept method)

Private Attributes

- int [_SocketId](#)
Socket identifier.

Additional Inherited Members

13.8.1 Detailed Description

Wrapper for sockets use in C++.

Definition at line 29 of file socket.h.

13.8.2 Constructor & Destructor Documentation

13.8.2.1 SOCKET::SOCKET ()

Socket constructor.

Definition at line 29 of file socket.cpp.

References `OBJECT::_ObjName`, `_SocketId`, `OBJECT::App()`, `APPLICATION::Console`, and `CONSOLE::LogInfo()`.

Referenced by `Accept()`.

13.8.2.2 SOCKET::~~SOCKET () [virtual]

Socket destructor.

Warning

A call to 'close()' indeed close the communication in only one direction

Todo Use the function `shutdown()` instead of `close()`

Definition at line 85 of file socket.cpp.

References `_SocketId`.

13.8.2.3 SOCKET::SOCKET (int *SocketId*) [private]

Socket constructor (reserved for accept method)

Parameters

<i>SocketId</i>	Already existing socket identifier (opened with accept)
-----------------	---

Definition at line 67 of file socket.cpp.

References `OBJECT::_ObjName`, `OBJECT::App()`, `APPLICATION::Console`, and `IsConnected()`.

13.8.3 Member Function Documentation

13.8.3.1 SOCKET & SOCKET::Accept ()

Accepts a connection (server mode)

Returns

Accepted socket

Definition at line 228 of file socket.cpp.

References `_SocketId`, `OBJECT::App()`, `APPLICATION::Console`, `CONSOLE::LogInfo()`, and `SOCKET()`.

Referenced by `APPLICATION::RunServer()`.

13.8.3.2 void SOCKET::Bind (unsigned short *PortNum*)

Binds the socket to local address (server mode)

Parameters

<i>PortNum</i>	Local port number
----------------	-------------------

Definition at line 150 of file socket.cpp.

References `_SocketId`, `OBJECT::App()`, `APPLICATION::Console`, and `CONSOLE::LogInfo()`.

Referenced by `APPLICATION::RunServer()`.

13.8.3.3 void SOCKET::Connect (std::string *IpAddr*, unsigned short *PortNum*)

Connect the socket to remote address (client mode)

Parameters

<i>IpAddr</i>	Remote IP address
<i>PortNum</i>	Remote port number

Definition at line 203 of file socket.cpp.

References `_SocketId`.

13.8.3.4 int SOCKET::GetId () const

Socket identifier getter.

Returns

Socket identifier

Definition at line 411 of file socket.cpp.

References `_SocketId`.

Referenced by `MANAGER::Create()`.

13.8.3.5 std::string SOCKET::GetLocalAddr () const

Local IP address getter.

Returns

Local IP address

Definition at line 102 of file socket.cpp.

References `_SocketId`.

Referenced by `CONNECTION::RunTask()`.

13.8.3.6 std::string SOCKET::GetRemoteAddr () const

Remote IP address getter.

Returns

Remote IP address

Definition at line 126 of file socket.cpp.

References `_SocketId`.

Referenced by `CONNECTION::RunTask()`.

13.8.3.7 bool SOCKET::IsConnected ()

Tells if the socket is connected.

Returns

true if connected
false if not connected

Definition at line 253 of file socket.cpp.

References `_SocketId`.

Referenced by `CONNECTION::RunTask()`, `SOCKET()`, and `CONNECTION::~~CONNECTION()`.

13.8.3.8 bool SOCKET::IsDataWaiting ()

Tells if data is waiting in the buffer.

Returns

true if data is waiting
false if no data is waiting

Bug `poll()` will not block as soon as the socket should block !

Todo Should use SIGIO instead of polling ? (Busy waiting against sleeping...)

Definition at line 279 of file socket.cpp.

References `_SocketId`.

Referenced by `CONNECTION::RunTask()`.

13.8.3.9 void SOCKET::Listen ()

Listens the socket (server mode)

Definition at line 182 of file socket.cpp.

References `_SocketId`, `OBJECT::App()`, `APPLICATION::Console`, and `CONSOLE::LogInfo()`.

Referenced by `APPLICATION::RunServer()`.

13.8.3.10 std::string SOCKET::Receive ()

Sends data to socket.

Returns

Received data

Definition at line 382 of file socket.cpp.

References `_SocketId`, `OBJECT::App()`, `APPLICATION::Console`, and `CONSOLE::LogInfo()`.

Referenced by `CONNECTION::RunTask()`.

13.8.3.11 void SOCKET::Send (const std::string & Data)

Sends data to socket.

Parameters

<i>Data</i>	Data to send
-------------	--------------

Definition at line 357 of file socket.cpp.

References `_SocketId`, `OBJECT::App()`, `APPLICATION::Console`, and `CONSOLE::LogInfo()`.

Referenced by `CONNECTION::RunTask()`, and `CONNECTION::~~CONNECTION()`.

13.8.3.12 `bool SOCKET::WaitData ()`

Definition at line 303 of file socket.cpp.

References `_SocketId`, `OBJECT::App()`, `APPLICATION::Console`, `CONSOLE::LogInfo()`, and `SOURCE_LINE`.

13.8.4 Member Data Documentation

13.8.4.1 `int SOCKET::_SocketId` `[private]`

Socket identifier.

Definition at line 165 of file socket.h.

Referenced by `Accept()`, `Bind()`, `Connect()`, `GetId()`, `GetLocalAddr()`, `GetRemoteAddr()`, `IsConnected()`, `IsDataWaiting()`, `Listen()`, `Receive()`, `Send()`, `SOCKET()`, `WaitData()`, and `~SOCKET()`.

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

- [inc/socket.h](#)
- [src/socket.cpp](#)

13.9 THREAD Class Reference

Wrapper for POSIX threads use in C++.

`#include <thread.h>`

Inherits [OBJECT](#).

Public Member Functions

- [THREAD](#) (`void *(*Proc)(void *)`, `void *Arg`)
Thread constructor.
- [~THREAD](#) ()
Thread destructor.
- `void` [Run](#) ()
Starts the thread.
- `void` [Cancel](#) ()
Stop the thread.

Static Private Member Functions

- `static void *` [ThreadFunction](#) (`void *Arg`)
Thread function.

Private Attributes

- pthread_t [_ThreadId](#)
POSIX thread identifier.
- pthread_attr_t [_Attr](#)
POSIX thread attributes.
- sigset_t [_SigMask](#)
POSIX signal thread mask.
- void (*)([_Procedure](#))(void *)
Pointer to the function the thread has to run.
- void * [_Argument](#)
Pointer to the argument the thread has to pass to its function.

Additional Inherited Members

13.9.1 Detailed Description

Wrapper for POSIX threads use in C++.

Definition at line 25 of file thread.h.

13.9.2 Constructor & Destructor Documentation

13.9.2.1 THREAD::THREAD (void (*)(void *) *Proc*, void * *Arg*)

Thread constructor.

Parameters

<i>Proc</i>	Pointer to function used to run thread
<i>Arg</i>	Argument to pass to the function

Definition at line 31 of file thread.cpp.

References [_Attr](#), [OBJECT::_ObjName](#), [_SigMask](#), [OBJECT::App\(\)](#), and [APPLICATION::Console](#).

13.9.2.2 THREAD::~~THREAD ()

Thread destructor.

Definition at line 66 of file thread.cpp.

References [_Attr](#), [OBJECT::_ObjName](#), [_ThreadId](#), [OBJECT::App\(\)](#), and [APPLICATION::Console](#).

13.9.3 Member Function Documentation

13.9.3.1 void THREAD::Cancel ()

Stop the thread.

Definition at line 106 of file thread.cpp.

References [_ThreadId](#).

Referenced by [CONNECTION::~~CONNECTION\(\)](#).

13.9.3.2 void THREAD::Run ()

Starts the thread.

Start the thread.

Definition at line 88 of file thread.cpp.

References `_Attr`, `_SigMask`, `_ThreadId`, and `ThreadFunction()`.

Referenced by `CONNECTION::CONNECTION()`.

13.9.3.3 void * THREAD::ThreadFunction (void * Arg) [static], [private]

Thread function.

Parameters

<i>Arg</i>	Generic pointer to current thread
------------	-----------------------------------

Returns

Generic pointer to get thread return

Definition at line 124 of file thread.cpp.

References `_Argument`, and `_Procedure`.

Referenced by `Run()`.

13.9.4 Member Data Documentation**13.9.4.1 void* THREAD::_Argument [private]**

Pointer to the argument the thread has to pass to its function.

Definition at line 87 of file thread.h.

Referenced by `ThreadFunction()`.

13.9.4.2 pthread_attr_t THREAD::_Attr [private]

POSIX thread attributes.

Definition at line 78 of file thread.h.

Referenced by `Run()`, `THREAD()`, and `~THREAD()`.

13.9.4.3 void*(* THREAD::_Procedure)(void *) [private]

Pointer to the function the thread has to run.

Definition at line 84 of file thread.h.

Referenced by `ThreadFunction()`.

13.9.4.4 sigset_t THREAD::_SigMask [private]

POSIX signal thread mask.

Definition at line 81 of file thread.h.

Referenced by `Run()`, and `THREAD()`.

13.9.4.5 pthread_t THREAD::_ThreadId [private]

POSIX thread identifier.

Definition at line 75 of file thread.h.

Referenced by Cancel(), Run(), and ~THREAD().

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

- [inc/thread.h](#)
- [src/thread.cpp](#)

Chapter 14

File Documentation

14.1 inc/application.h File Reference

Header for main application.

```
#include "object.h"
#include "parameters.h"
#include "console.h"
#include "manager.h"
```

Classes

- class [APPLICATION](#)
Running program and all its components as subobjects.

14.1.1 Detailed Description

Header for main application.

Author

Olivier de BLIC

Definition in file [application.h](#).

14.2 inc/connection.h File Reference

Header for host connection use.

```
#include "object.h"
#include "manager.h"
#include "socket.h"
#include "thread.h"
```

Classes

- class [CONNECTION](#)
Host connection with socket and thread embedded.

14.2.1 Detailed Description

Header for host connection use.

Author

Olivier de BLIC

Definition in file [connection.h](#).

14.3 inc/console.h File Reference

Header for console use.

```
#include <pthread.h>
#include <string>
#include <sstream>
#include "exception.h"
```

Classes

- class [CONSOLE](#)
Interface for console output and formatted logs.

Macros

- #define [SSTR](#)(NB) #NB
- #define [STR](#)(MSG) [SSTR](#)(MSG)
- #define [SOURCE_LINE](#) __FILE__ ":" [STR](#)(__LINE__)

Enumerations

- enum [LOG_TYPE](#) {
 [LOG_CTOR](#), [LOG_DTOR](#), [LOG_BLANK](#), [LOG_DEBUG](#),
 [LOG_INFO](#), [LOG_WARN](#), [LOG_ERROR](#), [LOG_EXCEPT](#),
 [LOG_SIGNAL](#) }
Enumeration of log types.

14.3.1 Detailed Description

Header for console use.

Author

Olivier de BLIC

Definition in file [console.h](#).

14.3.2 Macro Definition Documentation

14.3.2.1 #define SOURCE_LINE __FILE__ ":" STR(__LINE__)

Definition at line 25 of file console.h.

Referenced by [MANAGER::Add\(\)](#), [MANAGER::Remove\(\)](#), [CONNECTION::RunTask\(\)](#), and [SOCKET::WaitData\(\)](#).

14.3.2.2 `#define SSTR(NB) #NB`

Definition at line 23 of file console.h.

14.3.2.3 `#define STR(MSG) SSTR(MSG)`

Definition at line 24 of file console.h.

14.3.3 Enumeration Type Documentation

14.3.3.1 `enum LOG_TYPE`

Enumeration of log types.

Enumerator

`LOG_CTOR`
`LOG_DTOR`
`LOG_BLANK`
`LOG_DEBUG`
`LOG_INFO`
`LOG_WARN`
`LOG_ERROR`
`LOG_EXCEPT`
`LOG_SIGNAL`

Definition at line 30 of file console.h.

14.4 inc/exception.h File Reference

Header for exceptions use.

```
#include <exception>
#include <string>
```

Classes

- class [EXCEPTION](#)
Generic exception dedicated for the application.

14.4.1 Detailed Description

Header for exceptions use.

Author

Olivier de BLIC

Definition in file [exception.h](#).

14.5 inc/manager.h File Reference

Header for connections management.

```
#include <pthread.h>
#include <set>
#include "object.h"
```

Classes

- class [MANAGER](#)
Connections manager.

Typedefs

- typedef std::set< [CONNECTION](#) * > [CONTAINER](#)
Container for unqiues values set (pointers to host connections)

14.5.1 Detailed Description

Header for connections management.

Author

Olivier de BLIC

Definition in file [manager.h](#).

14.5.2 Typedef Documentation

14.5.2.1 typedef std::set<[CONNECTION](#)*> [CONTAINER](#)

Container for unqiues values set (pointers to host connections)

Definition at line 25 of file [manager.h](#).

14.6 inc/object.h File Reference

Header for objects using [APPLICATION](#).

```
#include <string>
```

Classes

- class [OBJECT](#)
Base class to give objects a name and an access to the object [APPLICATION](#).

14.6.1 Detailed Description

Header for objects using [APPLICATION](#).

Author

Olivier de BLIC

Definition in file [object.h](#).

14.7 inc/parameters.h File Reference

Header for parameters use.

```
#include <string>
```

Classes

- class [PARAMETERS](#)
Parser to read parameters passed to the [main\(\)](#) function.

14.7.1 Detailed Description

Header for parameters use.

Author

Olivier de BLIC

Definition in file [parameters.h](#).

14.8 inc/socket.h File Reference

Header for sockets use.

```
#include <sys/socket.h>
#include <string>
#include <arpa/inet.h>
#include <unistd.h>
#include "object.h"
```

Classes

- class [SOCKET](#)
Wrapper for sockets use in C++.

14.8.1 Detailed Description

Header for sockets use.

Author

Olivier de BLIC

Definition in file [socket.h](#).

14.9 inc/thread.h File Reference

Header for threads use.

```
#include <pthread.h>
#include "object.h"
```

Classes

- class [THREAD](#)

Wrapper for POSIX threads use in C++.

14.9.1 Detailed Description

Header for threads use.

Author

Olivier de BLIC

Definition in file [thread.h](#).

14.10 src/application.cpp File Reference

Module for main application.

```
#include <signal.h>
#include <stdlib.h>
#include <stdio.h>
#include <execinfo.h>
#include "application.h"
#include "object.h"
#include "socket.h"
#include "connection.h"
#include "exception.h"
```

14.10.1 Detailed Description

Module for main application.

Author

Olivier de BLIC

Definition in file [application.cpp](#).

14.11 src/connection.cpp File Reference

Module for host connection use.

```
#include <string>
#include <unistd.h>
#include "connection.h"
#include "object.h"
#include "application.h"
```

Macros

- `#define CYCLE_DURATION_MS (1000)`
- `#define SLICE_DURATION_MS (50)`
- `#define CYCLE_DURATION_US ((CYCLE_DURATION_MS) * (1000))`
- `#define SLICE_DURATION_US ((SLICE_DURATION_MS) * (1000))`

14.11.1 Detailed Description

Module for host connection use.

Author

Olivier de BLIC

Definition in file [connection.cpp](#).

14.11.2 Macro Definition Documentation

14.11.2.1 `#define CYCLE_DURATION_MS (1000)`

Definition at line 21 of file [connection.cpp](#).

14.11.2.2 `#define CYCLE_DURATION_US ((CYCLE_DURATION_MS) * (1000))`

Definition at line 23 of file [connection.cpp](#).

Referenced by [CONNECTION::RunTask\(\)](#).

14.11.2.3 `#define SLICE_DURATION_MS (50)`

Definition at line 22 of file [connection.cpp](#).

14.11.2.4 `#define SLICE_DURATION_US ((SLICE_DURATION_MS) * (1000))`

Definition at line 24 of file [connection.cpp](#).

Referenced by [CONNECTION::RunTask\(\)](#).

14.12 src/console.cpp File Reference

Module for console use.

```
#include <string>
#include <iostream>
#include <sstream>
#include <fstream>
#include <signal.h>
#include <typeinfo>
#include <time.h>
#include "console.h"
#include "exception.h"
```

Macros

- `#define CODE_GRA "0"`
- `#define CODE_RED "1"`
- `#define CODE_GRE "2"`
- `#define CODE_YEL "3"`
- `#define CODE_BLU "4"`
- `#define CODE_MAG "5"`
- `#define CODE_CYA "6"`
- `#define CODE_WHI "7"`
- `#define CODE_BLK "9"`
- `#define ESC_SEQ(FG_COL, BG_COL) "\e[1;3" FG_COL ";4" BG_COL "m"`
- `#define ESC_GRA ESC_SEQ(CODE_GRA, CODE_BLK)`
- `#define ESC_RED ESC_SEQ(CODE_RED, CODE_BLK)`
- `#define ESC_GRE ESC_SEQ(CODE_GRE, CODE_BLK)`
- `#define ESC_YEL ESC_SEQ(CODE_YEL, CODE_BLK)`
- `#define ESC_BLU ESC_SEQ(CODE_BLU, CODE_BLK)`
- `#define ESC_MAG ESC_SEQ(CODE_MAG, CODE_BLK)`
- `#define ESC_CYA ESC_SEQ(CODE_CYA, CODE_BLK)`
- `#define ESC_WHI ESC_SEQ(CODE_WHI, CODE_BLK)`
- `#define ESC_BLK ESC_SEQ(CODE_BLK, CODE_BLK)`
- `#define ESC_STD "\e[0m"`

14.12.1 Detailed Description

Module for console use.

Author

Olivier de BLIC

Definition in file [console.cpp](#).

14.12.2 Macro Definition Documentation

14.12.2.1 `#define CODE_BLK "9"`

Definition at line 33 of file [console.cpp](#).

14.12.2.2 `#define CODE_BLU "4"`

Definition at line 29 of file console.cpp.

14.12.2.3 `#define CODE_CYA "6"`

Definition at line 31 of file console.cpp.

14.12.2.4 `#define CODE_GRA "0"`

Definition at line 25 of file console.cpp.

14.12.2.5 `#define CODE_GRE "2"`

Definition at line 27 of file console.cpp.

14.12.2.6 `#define CODE_MAG "5"`

Definition at line 30 of file console.cpp.

14.12.2.7 `#define CODE_RED "1"`

Definition at line 26 of file console.cpp.

14.12.2.8 `#define CODE_WHI "7"`

Definition at line 32 of file console.cpp.

14.12.2.9 `#define CODE_YEL "3"`

Definition at line 28 of file console.cpp.

14.12.2.10 `#define ESC_BLK ESC_SEQ(CODE_BLK, CODE_BLK)`

Definition at line 45 of file console.cpp.

14.12.2.11 `#define ESC_BLU ESC_SEQ(CODE_BLU, CODE_BLK)`

Definition at line 41 of file console.cpp.

Referenced by `CONSOLE::ColBlu()`.

14.12.2.12 `#define ESC_CYA ESC_SEQ(CODE_CYA, CODE_BLK)`

Definition at line 43 of file console.cpp.

Referenced by `CONSOLE::ColCya()`.

14.12.2.13 `#define ESC_GRA ESC_SEQ(CODE_GRA, CODE_BLK)`

Definition at line 37 of file console.cpp.

Referenced by `CONSOLE::ColGra()`.

14.12.2.14 `#define ESC_GRE ESC_SEQ(CODE_GRE, CODE_BLK)`

Definition at line 39 of file console.cpp.

Referenced by `CONSOLE::ColGre()`.

14.12.2.15 `#define ESC_MAG ESC_SEQ(CODE_MAG, CODE_BLK)`

Definition at line 42 of file console.cpp.

Referenced by `CONSOLE::ColMag()`.

14.12.2.16 `#define ESC_RED ESC_SEQ(CODE_RED, CODE_BLK)`

Definition at line 38 of file console.cpp.

Referenced by `CONSOLE::ColRed()`.

14.12.2.17 `#define ESC_SEQ(FG_COL, BG_COL) "\e[1;3" FG_COL ";4" BG_COL "m"`

Definition at line 36 of file console.cpp.

14.12.2.18 `#define ESC_STD "\e[0m"`

Definition at line 46 of file console.cpp.

Referenced by `CONSOLE::ColStd()`.

14.12.2.19 `#define ESC_WHI ESC_SEQ(CODE_WHI, CODE_BLK)`

Definition at line 44 of file console.cpp.

Referenced by `CONSOLE::ColWhi()`.

14.12.2.20 `#define ESC_YEL ESC_SEQ(CODE_YEL, CODE_BLK)`

Definition at line 40 of file console.cpp.

Referenced by `CONSOLE::ColYel()`.

14.13 `src/exception.cpp` File Reference

Module for exceptions use.

```
#include <string>
#include "exception.h"
```


14.13.1 Detailed Description

Module for exceptions use.

Author

Olivier de BLIC

Definition in file [exception.cpp](#).

14.14 src/main.cpp File Reference

Module containing the [main\(\)](#) function.

```
#include "object.h"
#include "application.h"
```

Functions

- [int main](#) (int argc, char *argv[])
Entry point of the program.

14.14.1 Detailed Description

Module containing the [main\(\)](#) function.

Author

Olivier de BLIC

Definition in file [main.cpp](#).

14.14.2 Function Documentation

14.14.2.1 `int main (int argc, char * argv [])`

Entry point of the program.

Todo Apply the **const** rules in the whole project

Use the **namespace** `ss = sea_star`; to declare identifiers dedicated to the project

heading1 Client connection

Unique ID

Number of client sent by server

Client disconnection

Parameters

<i>argc</i>	Number of arguments
<i>argv</i>	Values of arguments

Returns

EXIT_SUCCESS if in case of success
EXIT_FAILURE if an error occurred

Remarks

This function is just an interface between the caller and the objet [APPLICATION](#) which is the core of the program.

Definition at line 44 of file main.cpp.

References [OBJECT::_AppPtr](#), and [APPLICATION::Main\(\)](#).

14.15 src/manager.cpp File Reference

Module for connections management.

```
#include <set>
#include <pthread.h>
#include "manager.h"
#include "application.h"
#include "connection.h"
#include "socket.h"
#include "console.h"
```

14.15.1 Detailed Description

Module for connections management.

Author

Olivier de BLIC

Definition in file [manager.cpp](#).

14.16 src/object.cpp File Reference

Module for objects using [APPLICATION](#).

```
#include <string>
#include "object.h"
```

14.16.1 Detailed Description

Module for objects using [APPLICATION](#).

Author

Olivier de BLIC

Definition in file [object.cpp](#).

14.17 src/parameters.cpp File Reference

Module for parameters use.

```
#include <stdlib.h>
#include <unistd.h>
#include <iostream>
#include "parameters.h"
#include "exception.h"
#include "application.h"
```

Macros

- `#define DEFLT_SERV_PORT 1101`

14.17.1 Detailed Description

Module for parameters use.

Author

Olivier de BLIC

Definition in file [parameters.cpp](#).

14.17.2 Macro Definition Documentation

14.17.2.1 `#define DEFLT_SERV_PORT 1101`

Definition at line 22 of file parameters.cpp.

14.18 src/socket.cpp File Reference

Module for sockets use.

```
#include <socket.h>
#include <poll.h>
#include <string.h>
#include <sstream>
#include "application.h"
#include "exception.h"
```

14.18.1 Detailed Description

Module for sockets use.

Author

Olivier de BLIC

Definition in file [socket.cpp](#).

14.19 src/thread.cpp File Reference

Module for threads use.

```
#include <pthread.h>
#include <signal.h>
#include "thread.h"
#include "exception.h"
#include "application.h"
```

14.19.1 Detailed Description

Module for threads use.

Author

Olivier de BLIC

Definition in file [thread.cpp](#).

Index

- ~APPLICATION
 - APPLICATION, [28](#)
- ~CONNECTION
 - CONNECTION, [31](#)
- ~CONSOLE
 - CONSOLE, [34](#)
- ~EXCEPTION
 - EXCEPTION, [41](#)
- ~MANAGER
 - MANAGER, [43](#)
- ~OBJECT
 - OBJECT, [46](#)
- ~PARAMETERS
 - PARAMETERS, [49](#)
- ~SOCKET
 - SOCKET, [53](#)
- ~THREAD
 - THREAD, [58](#)
- _AlreadyParsed
 - PARAMETERS, [51](#)
- _AppPtr
 - OBJECT, [48](#)
- _Argument
 - THREAD, [59](#)
- _Attr
 - THREAD, [59](#)
- _Buffer
 - CONSOLE, [40](#)
- _Colors
 - CONSOLE, [40](#)
 - PARAMETERS, [51](#)
- _Container
 - MANAGER, [45](#)
- _Description
 - EXCEPTION, [42](#)
- _FullPath
 - CONSOLE, [40](#)
- _Help
 - PARAMETERS, [51](#)
- _HostId
 - CONNECTION, [32](#)
- _Lock
 - CONSOLE, [40](#)
 - MANAGER, [45](#)
- _Manager
 - CONNECTION, [32](#)
- _ObjName
 - OBJECT, [48](#)
- _PortNum
 - PARAMETERS, [51](#)
- _Procedure
 - THREAD, [59](#)
- _Running
 - APPLICATION, [29](#)
- _SigMask
 - THREAD, [59](#)
- _Socket
 - CONNECTION, [32](#)
- _SocketId
 - SOCKET, [57](#)
- _Splashscreen
 - PARAMETERS, [51](#)
- _Stream
 - CONSOLE, [40](#)
- _Thread
 - CONNECTION, [32](#)
- _ThreadId
 - THREAD, [59](#)
- _Verbose
 - CONSOLE, [40](#)
 - PARAMETERS, [51](#)
- APPLICATION, [27](#)
 - ~APPLICATION, [28](#)
 - _Running, [29](#)
 - APPLICATION, [28](#)
 - APPLICATION, [28](#)
 - Console, [29](#)
 - Main, [28](#)
 - MANAGER, [45](#)
 - Manager, [30](#)
 - Param, [30](#)
 - PrintBackTrace, [28](#)
 - RunServer, [29](#)
 - SetSignalConfig, [29](#)
 - SignalHandler, [29](#)
- Accept
 - SOCKET, [53](#)
- Add
 - MANAGER, [43](#)
- App
 - OBJECT, [47](#)
- Bind
 - SOCKET, [53](#)
- CODE_BLK
 - console.cpp, [68](#)
- CODE_BLU

- console.cpp, 68
- CODE_CYA
 - console.cpp, 69
- CODE_GRA
 - console.cpp, 69
- CODE_GRE
 - console.cpp, 69
- CODE_MAG
 - console.cpp, 69
- CODE_RED
 - console.cpp, 69
- CODE_WHI
 - console.cpp, 69
- CODE_YEL
 - console.cpp, 69
- CONNECTION, 30
 - ~CONNECTION, 31
 - _HostId, 32
 - _Manager, 32
 - _Socket, 32
 - _Thread, 32
 - CONNECTION, 31
 - CONNECTION, 31
 - GetHostID, 31
 - RunTask, 31
- CONSOLE, 32
 - ~CONSOLE, 34
 - _Buffer, 40
 - _Colors, 40
 - _FullPath, 40
 - _Lock, 40
 - _Stream, 40
 - _Verbose, 40
 - CONSOLE, 34
 - ColBlu, 34
 - ColCya, 34
 - ColGra, 35
 - ColGre, 35
 - ColMag, 35
 - ColRed, 35
 - ColStd, 36
 - ColWhi, 36
 - ColYel, 36
 - CONSOLE, 34
 - GetTimeStamp, 36
 - InitLogger, 37
 - LogBlank, 37
 - LogError, 37
 - LogExcept, 37
 - LogInfo, 37
 - LogSignal, 38
 - LogWarn, 38
 - PrintHelp, 38
 - PrintLogLine, 38
 - PrintSplashScreen, 39
 - ReleaseLogger, 39
 - SetColors, 39
 - SetFullPath, 39
 - SetVerbose, 39
- CONTAINER
 - manager.h, 64
- CYCLE_DURATION_MS
 - connection.cpp, 67
- CYCLE_DURATION_US
 - connection.cpp, 67
- Cancel
 - THREAD, 58
- ColBlu
 - CONSOLE, 34
- ColCya
 - CONSOLE, 34
- ColGra
 - CONSOLE, 35
- ColGre
 - CONSOLE, 35
- ColMag
 - CONSOLE, 35
- ColRed
 - CONSOLE, 35
- ColStd
 - CONSOLE, 36
- ColWhi
 - CONSOLE, 36
- ColYel
 - CONSOLE, 36
- Connect
 - SOCKET, 55
- connection.cpp
 - CYCLE_DURATION_MS, 67
 - CYCLE_DURATION_US, 67
 - SLICE_DURATION_MS, 67
 - SLICE_DURATION_US, 67
- Console
 - APPLICATION, 29
- console.h
 - LOG_BLANK, 63
 - LOG_CTOR, 63
 - LOG_DEBUG, 63
 - LOG_DTOR, 63
 - LOG_ERROR, 63
 - LOG_EXCEPT, 63
 - LOG_INFO, 63
 - LOG_SIGNAL, 63
 - LOG_WARN, 63
- console.cpp
 - CODE_BLK, 68
 - CODE_BLU, 68
 - CODE_CYA, 69
 - CODE_GRA, 69
 - CODE_GRE, 69
 - CODE_MAG, 69
 - CODE_RED, 69
 - CODE_WHI, 69
 - CODE_YEL, 69
 - ESC_BLK, 69
 - ESC_BLU, 69

- ESC_CYA, [69](#)
- ESC_GRA, [69](#)
- ESC_GRE, [70](#)
- ESC_MAG, [70](#)
- ESC_RED, [70](#)
- ESC_SEQ, [70](#)
- ESC_STD, [70](#)
- ESC_WHI, [70](#)
- ESC_YEL, [70](#)
- console.h
 - LOG_TYPE, [63](#)
 - SOURCE_LINE, [62](#)
 - SSTR, [62](#)
 - STR, [63](#)
- Count
 - MANAGER, [44](#)
- Create
 - MANAGER, [44](#)
- DEFLT_SERV_PORT
 - parameters.cpp, [73](#)
- Destroy
 - MANAGER, [44](#)
- ESC_BLK
 - console.cpp, [69](#)
- ESC_BLU
 - console.cpp, [69](#)
- ESC_CYA
 - console.cpp, [69](#)
- ESC_GRA
 - console.cpp, [69](#)
- ESC_GRE
 - console.cpp, [70](#)
- ESC_MAG
 - console.cpp, [70](#)
- ESC_RED
 - console.cpp, [70](#)
- ESC_SEQ
 - console.cpp, [70](#)
- ESC_STD
 - console.cpp, [70](#)
- ESC_WHI
 - console.cpp, [70](#)
- ESC_YEL
 - console.cpp, [70](#)
- EXCEPTION, [40](#)
 - ~EXCEPTION, [41](#)
 - _Description, [42](#)
 - EXCEPTION, [41](#)
 - EXCEPTION, [41](#)
 - GetDescription, [41](#)
 - what, [42](#)
- GetColors
 - PARAMETERS, [49](#)
- GetDescription
 - EXCEPTION, [41](#)
- GetHelp
 - PARAMETERS, [49](#)
- GetHostID
 - CONNECTION, [31](#)
- GetId
 - SOCKET, [55](#)
- GetLocalAddr
 - SOCKET, [55](#)
- GetRemoteAddr
 - SOCKET, [55](#)
- GetServerPort
 - PARAMETERS, [49](#)
- GetSplashscreen
 - PARAMETERS, [50](#)
- GetTimeStamp
 - CONSOLE, [36](#)
- GetVerbose
 - PARAMETERS, [50](#)
- inc/application.h, [61](#)
- inc/connection.h, [61](#)
- inc/console.h, [62](#)
- inc/exception.h, [63](#)
- inc/manager.h, [64](#)
- inc/object.h, [64](#)
- inc/parameters.h, [65](#)
- inc/socket.h, [65](#)
- inc/thread.h, [66](#)
- InitLogger
 - CONSOLE, [37](#)
- IsConnected
 - SOCKET, [55](#)
- IsDataWaiting
 - SOCKET, [56](#)
- LOG_BLANK
 - console.h, [63](#)
- LOG_CTOR
 - console.h, [63](#)
- LOG_DEBUG
 - console.h, [63](#)
- LOG_DTOR
 - console.h, [63](#)
- LOG_ERROR
 - console.h, [63](#)
- LOG_EXCEPT
 - console.h, [63](#)
- LOG_INFO
 - console.h, [63](#)
- LOG_SIGNAL
 - console.h, [63](#)
- LOG_WARN
 - console.h, [63](#)
- LOG_TYPE
 - console.h, [63](#)
- Listen
 - SOCKET, [56](#)
- LogBlank
 - CONSOLE, [37](#)
- LogError

- CONSOLE, 37
- LogExcept
 - CONSOLE, 37
- LogInfo
 - CONSOLE, 37
- LogSignal
 - CONSOLE, 38
- LogWarn
 - CONSOLE, 38
- MANAGER, 42
 - ~MANAGER, 43
 - _Container, 45
 - _Lock, 45
 - APPLICATION, 45
 - Add, 43
 - Count, 44
 - Create, 44
 - Destroy, 44
 - MANAGER, 43
 - MANAGER, 43
 - NewHostID, 44
 - Remove, 45
- Main
 - APPLICATION, 28
- main
 - main.cpp, 71
 - OBJECT, 47
- main.cpp
 - main, 71
- Manager
 - APPLICATION, 30
- manager.h
 - CONTAINER, 64
- NewHostID
 - MANAGER, 44
- OBJECT, 45
 - ~OBJECT, 46
 - _AppPtr, 48
 - _ObjName, 48
 - App, 47
 - main, 47
 - OBJECT, 46
 - OBJECT, 46
- PARAMETERS, 48
 - ~PARAMETERS, 49
 - _AlreadyParsed, 51
 - _Colors, 51
 - _Help, 51
 - _PortNum, 51
 - _Splashscreen, 51
 - _Verbose, 51
 - GetColors, 49
 - GetHelp, 49
 - GetServerPort, 49
 - GetSplashscreen, 50
 - GetVerbose, 50
 - PARAMETERS, 49
 - PARAMETERS, 49
 - Parse, 50
- Param
 - APPLICATION, 30
- parameters.cpp
 - DEFLT_SERV_PORT, 73
- Parse
 - PARAMETERS, 50
- PrintBackTrace
 - APPLICATION, 28
- PrintHelp
 - CONSOLE, 38
- PrintLogLine
 - CONSOLE, 38
- PrintSplashScreen
 - CONSOLE, 39
- Receive
 - SOCKET, 56
- ReleaseLogger
 - CONSOLE, 39
- Remove
 - MANAGER, 45
- Run
 - THREAD, 58
- RunServer
 - APPLICATION, 29
- RunTask
 - CONNECTION, 31
- SLICE_DURATION_MS
 - connection.cpp, 67
- SLICE_DURATION_US
 - connection.cpp, 67
- SOCKET, 52
 - ~SOCKET, 53
 - _SocketId, 57
 - Accept, 53
 - Bind, 53
 - Connect, 55
 - GetId, 55
 - GetLocalAddr, 55
 - GetRemoteAddr, 55
 - IsConnected, 55
 - IsDataWaiting, 56
 - Listen, 56
 - Receive, 56
 - SOCKET, 53
 - Send, 56
 - SOCKET, 53
 - WaitData, 57
- SOURCE_LINE
 - console.h, 62
- SSTR
 - console.h, 62
- STR
 - console.h, 63

- Send
 - SOCKET, [56](#)
- SetColors
 - CONSOLE, [39](#)
- SetFullPath
 - CONSOLE, [39](#)
- SetSignalConfig
 - APPLICATION, [29](#)
- SetVerbose
 - CONSOLE, [39](#)
- SignalHandler
 - APPLICATION, [29](#)
- src/application.cpp, [66](#)
- src/connection.cpp, [67](#)
- src/console.cpp, [68](#)
- src/exception.cpp, [70](#)
- src/main.cpp, [71](#)
- src/manager.cpp, [72](#)
- src/object.cpp, [72](#)
- src/parameters.cpp, [73](#)
- src/socket.cpp, [73](#)
- src/thread.cpp, [74](#)
- THREAD, [57](#)
 - ~THREAD, [58](#)
 - _Argument, [59](#)
 - _Attr, [59](#)
 - _Procedure, [59](#)
 - _SigMask, [59](#)
 - _ThreadId, [59](#)
 - Cancel, [58](#)
 - Run, [58](#)
 - THREAD, [58](#)
 - THREAD, [58](#)
 - ThreadFunction, [59](#)
- ThreadFunction
 - THREAD, [59](#)
- WaitData
 - SOCKET, [57](#)
- what
 - EXCEPTION, [42](#)