

Network lib in C

Generated by Doxygen 1.10.0

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 log_meta_s Struct Reference	5
3.2 net_client_s Struct Reference	5
3.2.1 Detailed Description	5
3.3 net_server_s Struct Reference	6
3.3.1 Detailed Description	6
4 File Documentation	7
4.1 log.h	7
4.2 net.h	8
Index	11

Chapter 1

Class Index

1.1 Class List

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

log_meta_s	5
net_client_s		
	Représente un client réseau connecté au serveur	5
net_server_s		
	Contient les informations et états du serveur	6

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

include/ log.h	7
include/ net.h	8

Chapter 3

Class Documentation

3.1 log_meta_s Struct Reference

Public Attributes

- const char * **color**
- const char * **label**

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

- include/log.h

3.2 net_client_s Struct Reference

Représente un client réseau connecté au serveur.

```
#include <net.h>
```

Public Attributes

- int **fd**
- bool **active**
- char **buffer** [BUFFER_SIZE]

3.2.1 Detailed Description

Représente un client réseau connecté au serveur.

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

- include/net.h

3.3 net_server_s Struct Reference

Contient les informations et états du serveur.

```
#include <net.h>
```

Public Attributes

- int **listen_fd**
- unsigned int **port**
- struct pollfd **pfds** [MAX_CLIENTS]
- [net_client_t](#) **clients** [MAX_CLIENTS]
- bool **running**

3.3.1 Detailed Description

Contient les informations et états du serveur.

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

- include/net.h

Chapter 4

File Documentation

4.1 log.h

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** network_lib
00004 ** File description:
00005 ** log
00006 */
00007
00008 #ifndef LOG_H_
00009     #define LOG_H_
00010
00011     /* ===== MACROS ===== */
00012
00013     #define RED        "\033[1;31m"
00014     #define GREEN      "\033[1;32m"
00015     #define YELLOW     "\033[1;33m"
00016     #define BLUE       "\033[1;34m"
00017     #define RESET      "\033[0m"
00018
00019     #define LOG_INFO(...)  log_message(LOG_LEVEL_INFO, __VA_ARGS__)
00020     #define LOG_DEBUG(...) log_message(LOG_LEVEL_DEBUG, __VA_ARGS__)
00021     #define LOG_WARN(...)  log_message(LOG_LEVEL_WARN, __VA_ARGS__)
00022     #define LOG_ERROR(...) log_message(LOG_LEVEL_ERROR, __VA_ARGS__)
00023
00024     /* ===== INCLUDES ===== */
00025
00026     #include <stdio.h>
00027     #include <stdarg.h>
00028     #include <stddef.h>
00029     #include <string.h>
00030     #include <unistd.h>
00031     #include <sys/socket.h>
00032     #include <errno.h>
00033
00034     /* ===== ENUM ===== */
00035
00036     typedef enum log_level_e {
00037         LOG_LEVEL_INFO,
00038         LOG_LEVEL_DEBUG,
00039         LOG_LEVEL_WARN,
00040         LOG_LEVEL_ERROR,
00041     } log_level_t;
00042
00043     /* ===== STRUCT ===== */
00044
00045     typedef struct log_meta_s {
00046         const char *color;
00047         const char *label;
00048     } log_meta_t;
00049
00050     /* ===== UTILS ===== */
00051
00052     void log_message(log_level_t level, const char *fmt, ...);
00053
00054 #endif /* !LOG_H_ */
```

4.2 net.h

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** network_lib
00004 ** File description:
00005 ** net
00006 */
00007
00008 #ifndef NET_H_
00009     #define NET_H_
00010
00011     #define MAX_CLIENTS 128
00012     #define BUFFER_SIZE 1024
00013
00014     #include <unistd.h>
00015     #include <stdlib.h>
00016     #include <stdio.h>
00017     #include <stdbool.h>
00018     #include <string.h>
00019     #include <errno.h>
00020     #include <poll.h>
00021     #include <sys/socket.h>
00022     #include <netinet/in.h>
00023
00024
00025
00026     /* ===== STRUCT ===== */
00027
00031 typedef struct net_client_s {
00032     int fd; // Descripteur de socket du client
00033     bool active; // Indique si le client est actif
00034     char buffer[BUFFER_SIZE]; // Tampon de réception des données
00035 } net_client_t;
00036
00040 typedef struct net_server_s {
00041     int listen_fd; // Socket d'écoute
00042     unsigned int port; // Port TCP utilisé par le serveur
00043     struct pollfd pfd[MAX_CLIENTS]; // Tableau de pollfd pour la surveillance
00044     net_client_t clients[MAX_CLIENTS]; // Tableau des clients connectés
00045     bool running; // Indique si le serveur est en cours d'exécution
00046 } net_server_t;
00047
00048
00049
00050     /* ===== CORE ===== */
00051
00059 net_server_t *net_server_create(unsigned int port);
00060
00067 bool net_server_start(net_server_t *server);
00068
00074 void net_server_poll(net_server_t *server);
00075
00081 void net_server_stop(net_server_t *server);
00082
00088 void net_server_destroy(net_server_t *server);
00089
00090
00091
00092     /* ===== UTILS ===== */
00093
00100 void net_send(int fd, const char *msg);
00101
00108 void close_socket(char *msg, int socket);
00109
00110
00111
00112     /* ===== CLIENTS ===== */
00113
00119 void init_clients_array(net_server_t *server);
00120
00127 void net_close_client(net_server_t *server, int fd);
00128
00134 void net_close_all_clients(net_server_t *server);
00135
00136
00137
00138     /* ===== POLL ===== */
00139
00145 void init_poll_fds(net_server_t *server);
00146
00147
00148
00149     /* ===== HANDLERS ===== */
00150
00156 void handle_disconnect(int fd);
00157

```

```
00163 void handle_connect(int fd);
00164
00171 void handle_data(int fd, char *data);
00172
00173 #endif /* NET_H_ */
```


Index

include/log.h, [7](#)

include/net.h, [8](#)

log_meta_s, [5](#)

net_client_s, [5](#)

net_server_s, [6](#)