Network lib in C

Generated by Doxygen 1.10.0

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

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

```
/*
** EPITECH PROJECT, 2025
** network_lib
** File description:
** log
*/

#ifndef LOG_H_
    #define LOG_H_

            /* ================== MACROS ================== */

    #define RED     "\033[1;31m"
    #define GREEN   "\033[1;32m"
    #define YELLOW  "\033[1;33m"
    #define BLUE    "\033[1;34m"
    #define RESET   "\033[0m"

    #define LOG_INFO(...)  log_message(LOG_LEVEL_INFO, __VA_ARGS__)
    #define LOG_DEBUG(...) log_message(LOG_LEVEL_DEBUG, __VA_ARGS__)
    #define LOG_WARN(...)  log_message(LOG_LEVEL_WARN, __VA_ARGS__)
    #define LOG_ERROR(...) log_message(LOG_LEVEL_ERROR, __VA_ARGS__)

            /* ================== INCLUDES ================== */

    #include <stdio.h>
    #include <stdarg.h>
    #include <stddef.h>
    #include <string.h>
    #include <unistd.h>
    #include <sys/socket.h>
    #include <errno.h>

            /* ================== ENUM ================== */

typedef enum log_level_e {
    LOG_LEVEL_INFO,
    LOG_LEVEL_DEBUG,
    LOG_LEVEL_WARN,
    LOG_LEVEL_ERROR,
} log_level_t;

            /* ================== STRUCT ================== */

typedef struct log_meta_s {
    const char *color;
    const char *label;
} log_meta_t;

            /* ================== UTILS ================== */

void log_message(log_level_t level, const char *fmt, ...);

#endif /* !LOG_H_ */
```

4.2 net.h

```
/*
** EPITECH PROJECT, 2025
** network_lib
** File description:
** net
*/

#ifndef NET_H_
    #define NET_H_

    #define MAX_CLIENTS 128
    #define BUFFER_SIZE 1024

    #include <unistd.h>
    #include <stdlib.h>
    #include <stdio.h>
    #include <stdbool.h>
    #include <string.h>
    #include <errno.h>
    #include <poll.h>
    #include <sys/socket.h>
    #include <netinet/in.h>



            /* ================== STRUCT ================== */

typedef struct net_client_s {
    int fd;                             // Descripteur de socket du client
    bool active;                        // Indique si le client est actif
    char buffer[BUFFER_SIZE];           // Tampon de réception des données
} net_client_t;

typedef struct net_server_s {
    int listen_fd; // Socket d'écoute
    unsigned int port; // Port TCP utilisé par le serveur
    struct pollfd pfds[MAX_CLIENTS]; // Tableau de pollfd pour la surveillance
    net_client_t clients[MAX_CLIENTS]; // Tableau des clients connectés
    bool running; // Indique si le serveur est en cours d'exécution
} net_server_t;



            /* ================== CORE ================== */

net_server_t *net_server_create(unsigned int port);

bool net_server_start(net_server_t *server);

void net_server_poll(net_server_t *server);

void net_server_stop(net_server_t *server);

void net_server_destroy(net_server_t *server);



            /* ================== UTILS ================== */

void net_send(int fd, const char *msg);

void close_socket(char *msg, int socket);



            /* ================== CLIENTS ================== */

void init_clients_array(net_server_t *server);

void net_close_client(net_server_t *server, int fd);

void net_close_all_clients(net_server_t *server);



            /* ================== POLL ================== */

void init_poll_fds(net_server_t *server);



            /* ================== HANDLERS ================== */

void handle_disconnect(int fd);

void handle_connect(int fd);

void handle_data(int fd, char *data);

#endif /* NET_H_ */
```

Index