# ZipZop

Generated by Doxygen 1.8.13

# Contents

# Chapter 1

# Data Structure Index

## 1.1 Data Structures

Here are the data structures with brief descriptions:

# Chapter 2

# File Index

## 2.1 File List

Here is a list of all files with brief descriptions:

# Chapter 3

# Data Structure Documentation

## 3.1 client Struct Reference

### Data Fields

- const char * name
- int sockfd
- pthread_t thread

### 3.1.1 Field Documentation

#### 3.1.1.1 name

```
const char* client::name
```

#### 3.1.1.2 sockfd

```
int client::sockfd
```

#### 3.1.1.3 thread

```
pthread_t client::thread
```

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

- src/client.c

## 3.2 message Struct Reference

Struct representing a messege sent by some sender.

### Data Fields

- const char * content
- const char * sender_name

### 3.2.1 Detailed Description

Struct representing a messege sent by some sender.

### 3.2.2 Field Documentation

#### 3.2.2.1 content

```
const char* message::content
```

The content of the message

#### 3.2.2.2 sender_name

```
const char* message::sender_name
```

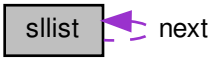
The username of the sender

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

- src/message.c

## 3.3 sllist Struct Reference

Collaboration diagram for sllist:



## Data Fields

- void * key
- struct sllist * next

### 3.3.1 Field Documentation

#### 3.3.1.1 key

```
void* sllist::key
```

#### 3.3.1.2 next

```
struct sllist* sllist::next
```

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

- src/sllist.c

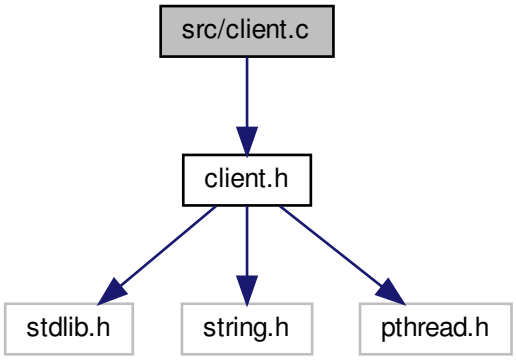
# Chapter 4

# File Documentation

## 4.1 src/client.c File Reference

```
#include "client.h"
```

Include dependency graph for client.c:




### Data Structures

- struct client

### Functions

- struct client * client_create (const char *name, int sockfd)
- void client_destroy (struct client *c)
- const char * client_get_name (struct client *c)
- int client_get_socket (struct client *c)
- pthread_t * client_get_thread (struct client *c)
- void client_set_name (struct client *c, const char *name)
- void client_set_socket (struct client *c, int sockfd)
- void client_set_thread (struct client *c, pthread_t thread)

### 4.1.1 Function Documentation

#### 4.1.1.1 client_create()

```
struct client* client_create (
            const char * name,
            int sockfd )
```

#### 4.1.1.2 client_destroy()

```
void client_destroy (
            struct client * c )
```

#### 4.1.1.3 client_get_name()

```
const char* client_get_name (
            struct client * c )
```

#### 4.1.1.4 client_get_socket()

```
int client_get_socket (
            struct client * c )
```

#### 4.1.1.5 client_get_thread()

```
pthread_t* client_get_thread (
            struct client * c )
```

#### 4.1.1.6 client_set_name()

```
void client_set_name (
            struct client * c,
            const char * name )
```

#### 4.1.1.7 client_set_socket()

```
void client_set_socket (
            struct client * c,
            int sockfd )
```

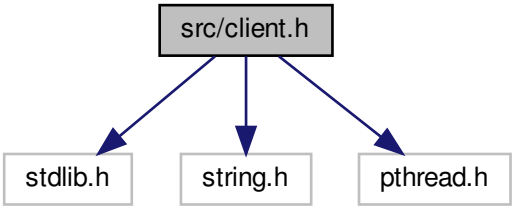
#### 4.1.1.8 client_set_thread()

```
void client_set_thread (
            struct client * c,
            pthread_t thread )
```

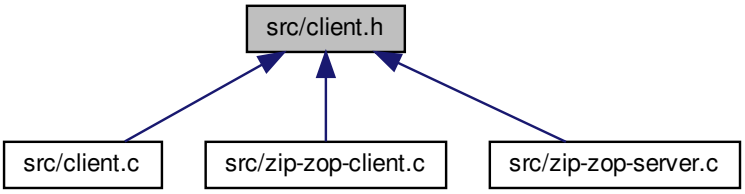
## 4.2 src/client.h File Reference

```
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
```

Include dependency graph for client.h:




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



### Functions

- struct client * client_create (const char *name, int sockfd)
- void client_destroy (struct client *c)
- const char * client_get_name (struct client *c)
- int client_get_socket (struct client *c)
- pthread_t * client_get_thread (struct client *c)
- void client_set_name (struct client *c, const char *name)
- void client_set_socket (struct client *c, int sockfd)
- void client_set_thread (struct client *c, pthread_t thread)

## 4.2.1 Function Documentation

#### 4.2.1.1 client_create()

```
struct client* client_create (
            const char * name,
            int sockfd )
```

#### 4.2.1.2 client_destroy()

```
void client_destroy (
            struct client * c )
```

#### 4.2.1.3 client_get_name()

```
const char* client_get_name (
            struct client * c )
```

#### 4.2.1.4 client_get_socket()

```
int client_get_socket (
            struct client * c )
```

#### 4.2.1.5 client_get_thread()

```
pthread_t* client_get_thread (
            struct client * c )
```

#### 4.2.1.6 client_set_name()

```
void client_set_name (
            struct client * c,
            const char * name )
```

#### 4.2.1.7 client_set_socket()

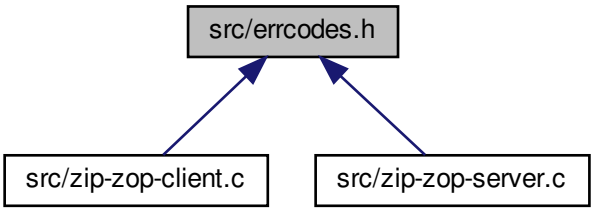
```
void client_set_socket (
            struct client * c,
            int sockfd )
```

#### 4.2.1.8 client_set_thread()

```
void client_set_thread (
            struct client * c,
            pthread_t thread )
```

## 4.3 src/errcodes.h File Reference

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




### Enumerations

- enum errcodes {
  E_SUCCESS, E_GETADDRINFO, E_BIND, E_LISTEN,
  E_BAD_ARGS, E_CONNECT, E_PTHREAD_CREATE }

### 4.3.1 Enumeration Type Documentation

#### 4.3.1.1 errcodes

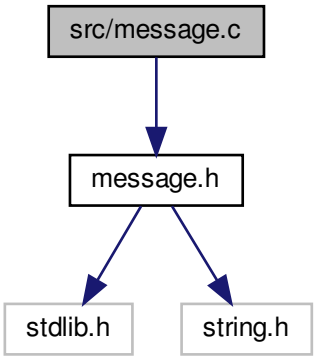
```
enum errcodes
```

**Enumerator**

| | |
|---|---|
| E_SUCCESS | |
| E_GETADDRINFO | |
| E_BIND | |
| E_LISTEN | |
| E_BAD_ARGS | |
| E_CONNECT | |
| E_PTHREAD_CREATE | |

## 4.4 src/message.c File Reference

`#include "message.h"`
Include dependency graph for message.c:




### Data Structures

- struct message

  *Struct representing a messege sent by some sender.*

### Functions

- struct message * message_create (const char *content, const char *sender_name)

  *Creates a message.*
- void message_destroy (struct message *m)

  *Destroys a message.*
- const char * message_get_content (struct message *m)

  *Get the message content.*
- const char * message_get_sender (struct message *m)

  *Get the message sender.*
- char * message_pack (struct message *m, int *len)

  *Serialize a message.*
- struct message * message_unpack (char *pack)

### 4.4.1 Function Documentation

#### 4.4.1.1 message_create()

```
struct message* message_create (
            const char * content,
            const char * sender_name )
```

Creates a message.

Both parameters will be copied into the message, so the user is free to `free()` the parameters passed to this function if necessary.

**Parameters**

| | | |
|---|---|---|
| in | *content* | The content of the message. |
| in | *sender_name* | The username of the sender. |

**Returns**

A pointer to a struct message in case of success, NULL otherwise. The message must be freed, using message_destroy(), when is not needed anymore.

**See also**

message_destroy

#### 4.4.1.2 message_destroy()

```
void message_destroy (
            struct message * m )
```

Destroys a message.

**Parameters**

| | | |
|---|---|---|
| in | *m* | A pointer to the message. |

**See also**

message_create

#### 4.4.1.3 message_get_content()

```
const char* message_get_content (
            struct message * m )
```

Get the message content.

**Parameters**

| | | |
|---|---|---|
| in | *m* | A pointer to the message. |

**Returns**

A pointer to the message content.

**Warning**

The returned value should not be freed.

#### 4.4.1.4 message_get_sender()

```
const char* message_get_sender (
            struct message * m )
```

Get the message sender.

**Parameters**

| | | |
|---|---|---|
| in | *m* | A pointer to the message. |

**Returns**

A pointer to the sender name.

**Warning**

The returned value should not be freed.

#### 4.4.1.5 message_pack()

```
char* message_pack (
            struct message * m,
            int * len )
```

Serialize a message.

Pack/Serialize the struct message in a format that can be sent through the network.

**Parameters**

| | | |
|---|---|---|
| in | *m* | A pointer to the message. |
| out | *len* | A pointer to a integer where the length of the serialized message will be stored. |

**Returns**

A pointer to the serialized message. This should be freed when is not necessary anymore.

**See also**

message_unpack

#### 4.4.1.6 message_unpack()

```
struct message* message_unpack (
            char * pack )
```

Deserialize a message.

Unpack/Deserialize a string into a struct message.

**Parameters**

| | | |
|---|---|---|
| in | *pack* | The string that represent the packed message generated by message_pack(). |

**Returns**

A pointer to the deserialized message. This should be freed when is not necessary anymore.

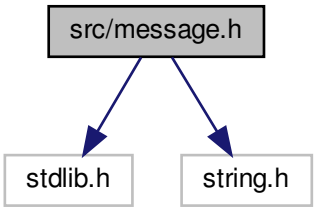
**See also**

message_pack

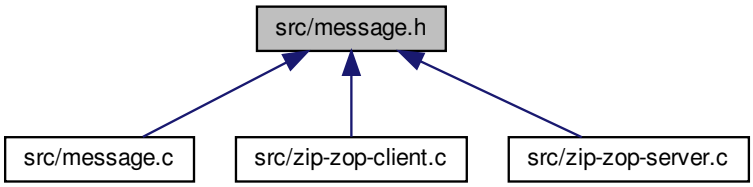
## 4.5 src/message.h File Reference

```
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for message.h:




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




## Functions

- struct message * message_create (const char *content, const char *sender_name)

  *Creates a message.*
- void message_destroy (struct message *m)

  *Destroys a message.*
- const char * message_get_content (struct message *m)

  *Get the message content.*
- const char * message_get_sender (struct message *m)

  *Get the message sender.*
- char * message_pack (struct message *m, int *len)

  *Serialize a message.*
- struct message * message_unpack (char *pack)

### 4.5.1 Function Documentation

#### 4.5.1.1 message_create()

```
struct message* message_create (
            const char * content,
            const char * sender_name )
```

Creates a message.

Both parameters will be copied into the message, so the user is free to `free()` the parameters passed to this function if necessary.

**Parameters**

| | | |
|---|---|---|
| in | *content* | The content of the message. |
| in | *sender_name* | The username of the sender. |

**Returns**

A pointer to a struct message in case of success, NULL otherwise. The message must be freed, using message_destroy(), when is not needed anymore.

**See also**

message_destroy

#### 4.5.1.2 message_destroy()

```
void message_destroy (
            struct message * m )
```

Destroys a message.

**Parameters**

| | | |
|---|---|---|
| in | *m* | A pointer to the message. |

**See also**

message_create

#### 4.5.1.3 message_get_content()

```
const char* message_get_content (
            struct message * m )
```

Get the message content.

**Parameters**

| | | |
|---|---|---|
| in | *m* | A pointer to the message. |

**Returns**

A pointer to the message content.

**Warning**

The returned value should not be freed.

#### 4.5.1.4 message_get_sender()

```
const char* message_get_sender (
            struct message * m )
```

Get the message sender.

**Parameters**

| | | |
|---|---|---|
| in | *m* | A pointer to the message. |

**Returns**

A pointer to the sender name.

**Warning**

The returned value should not be freed.

#### 4.5.1.5 message_pack()

```
char* message_pack (
            struct message * m,
            int * len )
```

Serialize a message.

Pack/Serialize the struct message in a format that can be sent through the network.

**Parameters**

| | | |
|---|---|---|
| in | *m* | A pointer to the message. |
| out | *len* | A pointer to a integer where the length of the serialized message will be stored. |

**Returns**

A pointer to the serialized message. This should be freed when is not necessary anymore.

**See also**

message_unpack

#### 4.5.1.6 message_unpack()

```
struct message* message_unpack (
            char * pack )
```

Deserialize a message.

Unpack/Deserialize a string into a struct message.

**Parameters**

| | | |
|---|---|---|
| in | *pack* | The string that represent the packed message generated by message_pack(). |

**Returns**

A pointer to the deserialized message. This should be freed when is not necessary anymore.

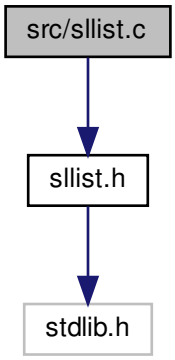
**See also**

message_pack

## 4.6 src/sllist.c File Reference

```
#include "sllist.h"
```

Include dependency graph for sllist.c:




## Data Structures

- struct sllist

## Functions

- struct sllist * sll_init (void)
- struct sllist * sll_get_next (struct sllist **l)
- void sll_insert_first (struct sllist **l, void *a)
- void sll_insert_last (struct sllist **l, void *a)
- void * sll_remove_first (struct sllist **l)
- void * sll_remove_last (struct sllist **l)
- void * sll_remove_elm (struct sllist **l, void *elm)
- void * sll_get_key (struct sllist *l)

### 4.6.1 Function Documentation

#### 4.6.1.1 sll_get_key()

```
void* sll_get_key (
            struct sllist * l )
```

#### 4.6.1.2 sll_get_next()

```
struct sllist* sll_get_next (
            struct sllist ** l )
```

#### 4.6.1.3 sll_init()

```
struct sllist* sll_init (
            void  )
```

#### 4.6.1.4 sll_insert_first()

```
void sll_insert_first (
            struct sllist ** l,
            void * a )
```

#### 4.6.1.5 sll_insert_last()

```
void sll_insert_last (
            struct sllist ** l,
            void * a )
```

#### 4.6.1.6 sll_remove_elm()

```
void* sll_remove_elm (
            struct sllist ** l,
            void * elm )
```

#### 4.6.1.7 sll_remove_first()

```
void* sll_remove_first (
            struct sllist ** l )
```

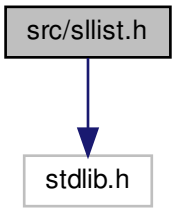
#### 4.6.1.8 sll_remove_last()

```
void* sll_remove_last (
            struct sllist ** l )
```

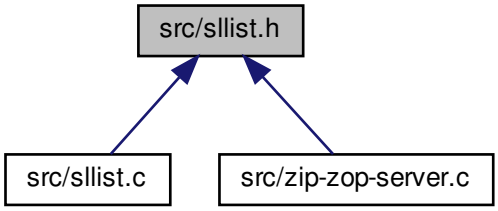
## 4.7 src/sllist.h File Reference

```
#include <stdlib.h>
```

Include dependency graph for sllist.h:




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




### Macros

- #define SLL_INIT() NULL;

### Functions

- struct sllist * sll_init (void)
- struct sllist * sll_get_next (struct sllist **l)
- void sll_insert_first (struct sllist **l, void *a)
- void sll_insert_last (struct sllist **l, void *a)
- void * sll_remove_first (struct sllist **l)
- void * sll_remove_last (struct sllist **l)
- void * sll_remove_elm (struct sllist **l, void *elm)
- void * sll_get_key (struct sllist *l)

### 4.7.1 Macro Definition Documentation

#### 4.7.1.1 SLL_INIT

```
#define SLL_INIT( ) NULL;
```

### 4.7.2 Function Documentation

#### 4.7.2.1 sll_get_key()

```
void* sll_get_key (
            struct sllist * l )
```

#### 4.7.2.2 sll_get_next()

```
struct sllist* sll_get_next (
            struct sllist ** l )
```

#### 4.7.2.3 sll_init()

```
struct sllist* sll_init (
            void  )
```

#### 4.7.2.4 sll_insert_first()

```
void sll_insert_first (
            struct sllist ** l,
            void * a )
```

#### 4.7.2.5 sll_insert_last()

```
void sll_insert_last (
            struct sllist ** l,
            void * a )
```

#### 4.7.2.6 sll_remove_elm()

```
void* sll_remove_elm (
            struct sllist ** l,
            void * elm )
```

#### 4.7.2.7 sll_remove_first()

```
void* sll_remove_first (
            struct sllist ** l )
```

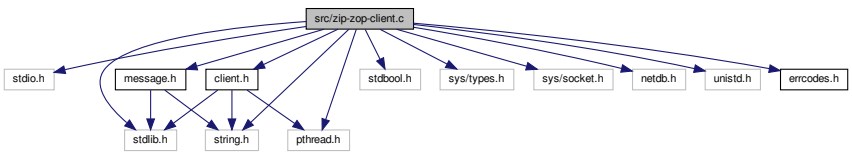
#### 4.7.2.8 sll_remove_last()

```
void* sll_remove_last (
            struct sllist ** l )
```

## 4.8 src/zip-zop-client.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <unistd.h>
#include <pthread.h>
#include "errcodes.h"
#include "message.h"
#include "client.h"
```

Include dependency graph for zip-zop-client.c:




### Macros

- #define PORT "1234"
- #define MESSAGE_LEN 2000

## Functions

- bool check_args (int argc)
- void print_usage (const char *name)
- void show_message (struct message *m)
- void * listen_thread (void *client)
- void * speak_thread (void *client)
- struct addrinfo * get_server_addr (const char *server_name)
- int create_and_connect (struct addrinfo *addr)
- void server_introduction (struct client *c)
- void communicate (const char *user_name, int sockfd)
- int main (int argc, char **argv)

### 4.8.1 Macro Definition Documentation

#### 4.8.1.1 MESSAGE_LEN

```
#define MESSAGE_LEN 2000
```

#### 4.8.1.2 PORT

```
#define PORT "1234"
```

### 4.8.2 Function Documentation

#### 4.8.2.1 check_args()

```
bool check_args (
            int argc )
```

#### 4.8.2.2 communicate()

```
void communicate (
            const char * user_name,
            int sockfd )
```

#### 4.8.2.3 create_and_connect()

```
int create_and_connect (
            struct addrinfo * addr )
```

#### 4.8.2.4 get_server_addr()

```
struct addrinfo* get_server_addr (
            const char * server_name )
```

#### 4.8.2.5 listen_thread()

```
void* listen_thread (
            void * client )
```

#### 4.8.2.6 main()

```
int main (
            int argc,
            char ** argv )
```

#### 4.8.2.7 print_usage()

```
void print_usage (
            const char * name )
```

#### 4.8.2.8 server_introduction()

```
void server_introduction (
            struct client * c )
```

#### 4.8.2.9 show_message()

```
void show_message (
            struct message * m )
```

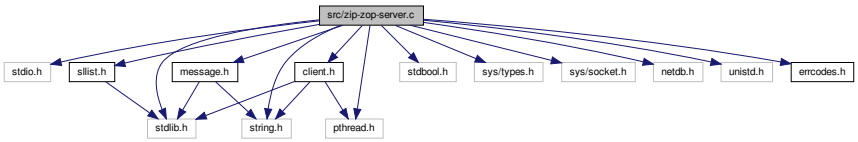
#### 4.8.2.10 speak_thread()

```
void* speak_thread (
            void * client )
```

## 4.9 src/zip-zop-server.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <unistd.h>
#include <pthread.h>
#include "errcodes.h"
#include "message.h"
#include "client.h"
#include "sllist.h"
```

Include dependency graph for zip-zop-server.c:




### Macros

- #define PORT "1234"
- #define BACKLOG 10
- #define CLIENT_NAME_LEN 100
- #define MESSAGE_LEN 2000

### Functions

- void client_thread_broadcast (struct client ∗c, const char ∗msg)
- void kill_client (struct client ∗c)
- void ∗ client_thread_listen (void ∗client)
- struct addrinfo ∗ get_internet_addr (void)
- int create_and_bind (struct addrinfo ∗addr)
- void create_new_client (int sockfd)
- int accept_clients (int sockfd)
- int main (void)

### Variables

- struct sllist * CLIENT_LIST = SLL_INIT()
- pthread_mutex_t CLIENT_LIST_MUTEX

## 4.9.1 Macro Definition Documentation

#### 4.9.1.1 BACKLOG

```
#define BACKLOG 10
```

#### 4.9.1.2 CLIENT_NAME_LEN

```
#define CLIENT_NAME_LEN 100
```

#### 4.9.1.3 MESSAGE_LEN

```
#define MESSAGE_LEN 2000
```

#### 4.9.1.4 PORT

```
#define PORT "1234"
```

## 4.9.2 Function Documentation

#### 4.9.2.1 accept_clients()

```
int accept_clients (
            int sockfd )
```

#### 4.9.2.2 client_thread_broadcast()

```
void client_thread_broadcast (
             struct client * c,
             const char * msg )
```

#### 4.9.2.3 client_thread_listen()

```
void* client_thread_listen (
             void * client )
```

#### 4.9.2.4 create_and_bind()

```
int create_and_bind (
             struct addrinfo * addr )
```

#### 4.9.2.5 create_new_client()

```
void create_new_client (
             int sockfd )
```

#### 4.9.2.6 get_internet_addr()

```
struct addrinfo* get_internet_addr (
             void  )
```

#### 4.9.2.7 kill_client()

```
void kill_client (
             struct client * c )
```

#### 4.9.2.8 main()

```
int main (
             void  )
```

### 4.9.3 Variable Documentation

#### 4.9.3.1 CLIENT_LIST

```
struct sllist* CLIENT_LIST = SLL_INIT()
```

#### 4.9.3.2 CLIENT_LIST_MUTEX

```
pthread_mutex_t CLIENT_LIST_MUTEX
```

# Index