

TCP client/server

Generated by Doxygen 1.9.4



---

<b>1 File Index</b>	<b>1</b>
1.1 File List . . . . .	1
<b>2 File Documentation</b>	<b>3</b>
2.1 tcp.h File Reference . . . . .	3
2.1.1 Detailed Description . . . . .	4
2.1.2 Macro Definition Documentation . . . . .	4
2.1.2.1 DEFAULT_IP . . . . .	4
2.1.2.2 DEFAULT_PORT . . . . .	4
2.1.2.3 DEFAULT_PROTOCOL . . . . .	4
2.1.2.4 MAX_BUFFER_SIZE . . . . .	4
2.1.2.5 MAX_UNSIGNED_SHORT_VAL . . . . .	4
2.1.2.6 RECV_FLAGS . . . . .	4
2.1.2.7 SEND_FLAGS . . . . .	4
2.1.3 Enumeration Type Documentation . . . . .	4
2.1.3.1 CLI_ARGS_COUNT . . . . .	4
2.1.3.2 CLIENT_ERRORS . . . . .	5
2.1.4 Function Documentation . . . . .	5
2.1.4.1 client_cli() . . . . .	6
2.1.4.2 client_connect() . . . . .	6
2.1.4.3 server_cli() . . . . .	6
2.1.4.4 server_start() . . . . .	7
2.2 tcp.h . . . . .	7
<b>Index</b>	<b>9</b>



# Chapter 1

## File Index

### 1.1 File List

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

<a href="#">tcp.h</a>	The header file include all functions used in program and some enums . . . . .	<a href="#">3</a>
-----------------------	--	-------------------



## Chapter 2

# File Documentation

### 2.1 tcp.h File Reference

The header file include all functions used in program and some enums.

```
#include <stdio.h>
```

#### Macros

- `#define MAX_BUFFER_SIZE 1024`
- `#define MAX_UNSIGNED_SHORT_VAL 65535`
- `#define DEFAULT_PROTOCOL 0`
- `#define SEND_FLAGS 0`
- `#define RECV_FLAGS 0`
- `#define DEFAULT_PORT 80`
- `#define DEFAULT_IP INADDR_LOOPBACK`

#### Enumerations

- `enum CLIENT_ERRORS {`  
    `INCORRECT_ARG_NUM = -5 , SERVER_ERROR = -4 , INCORRECT_PORT = -3 , INCORRECT_IP = -2 ,`  
    `CLIENT_CONNECT_ERROR = -1 , INCORRECT_SOCKET = -1 }`
- `enum CLI_ARGS_COUNT {`  
    `USR_CLI_DEFAULT_IP_N_PORT = 2 , USR_CLI_DEFAULT_PORT = 3 , USR_ALL_ARGS_PASSED = 4 ,`  
    `SERVER_CLI_DEFAULT_PORT = 2 ,`  
    `SERVER_ALL_ARGS_PASSED = 3 }`

#### Functions

- `int client_connect (const int ip_addr, const int port)`  
    *function that use TCP protocol to connect to network with passed ip and port*
- `int client_cli (const int argc, char *argv[])`  
    *function that handle command line arguments and connect to network by using `client_connect()` if arguments is correct.*
- `int server_start (int port)`  
    *start listen on chosen port through TCP protocol*
- `int server_cli (int argc, char *argv[])`  
    *function that handle command line arguments and start server by using `server_start()` if argument is correct*
- `void write_help ()`  
    *write help if help flag is used (-help)*

### 2.1.1 Detailed Description

The header file include all functions used in program and some enums.

### 2.1.2 Macro Definition Documentation

#### 2.1.2.1 DEFAULT\_IP

```
#define DEFAULT_IP INADDR_LOOPBACK
```

default ip. INADDR\_LOOPBACK is localhost ip.

#### 2.1.2.2 DEFAULT\_PORT

```
#define DEFAULT_PORT 80
```

default port value

#### 2.1.2.3 DEFAULT\_PROTOCOL

```
#define DEFAULT_PROTOCOL 0
```

used as third argument of socket() function

#### 2.1.2.4 MAX\_BUFFER\_SIZE

```
#define MAX_BUFFER_SIZE 1024
```

max buffer size that server will receive by using recv() function

#### 2.1.2.5 MAX\_UNSIGNED\_SHORT\_VAL

```
#define MAX_UNSIGNED_SHORT_VAL 65535
```

max unsigned short type value is  $2^{16} - 1 = 65535$

#### 2.1.2.6 RECV\_FLAGS

```
#define RECV_FLAGS 0
```

flags used in recv() function

#### 2.1.2.7 SEND\_FLAGS

```
#define SEND_FLAGS 0
```

flags used in send() function

### 2.1.3 Enumeration Type Documentation

#### 2.1.3.1 CLI\_ARGS\_COUNT

```
enum CLI_ARGS_COUNT
```



## Enumerator

USR_CLI_DEFAULT_IP_N_PORT	the value 2 is argument count passed through command line. If the program work like a client, that it require 4 arguments (first argument - the executable name, second is flag to indentify, that program working as a client (-client), third - ip address, fourth - port. But last two arguments can be omitted. The default ip address 127.0.0.1 (localhost), default port - 80.
USR_CLI_DEFAULT_PORT	3 arguments passed, the only missing one in port. Default port is 80.  See also <a href="#">USR_CLI_DEFAULT_IP_N_PORT</a>
USR_ALL_ARGS_PASSED	4 arguments passed, including ip address and port number.  See also <a href="#">USR_CLI_DEFAULT_IP_N_PORT</a>
SERVER_CLI_DEFAULT_PORT	2 arguments passed. If the program work as a server, that it require 3 arguments: first one is executable name, second is flag to indentify, that program working as a server (-server), third - port number. But last one can be ommited. The port will be set to default value - 80.
SERVER_ALL_ARGS_PASSED	3 arguments passed, including port number.  See also <a href="#">SERVER_CLI_DEFAULT_PORT</a>

## 2.1.3.2 CLIENT\_ERRORS

enum [CLIENT\\_ERRORS](#)

## Enumerator

INCORRECT_ARG_NUM	return value of <a href="#">client_cli()</a> or <a href="#">server_cli()</a> if incorrect number of arguments was passed
SERVER_ERROR	return value of <a href="#">server_start()</a> if error was detected
INCORRECT_PORT	return value of <a href="#">client_cli()</a> if incorrect port was passed
INCORRECT_IP	return value of <a href="#">client_cli()</a> if incorrect ip was passed
CLIENT_CONNECT_ERROR	code of client error that will returned in <a href="#">client_connect()</a> function
INCORRECT_SOCKET	if socket() function get error it return -1

## 2.1.4 Function Documentation

#### 2.1.4.1 client\_cli()

```
int client_cli (
    const int argc,
    char * argv[] )
```

function that handle command line arguments and connect to network by using [client\\_connect\(\)](#) if arguments is correct.

##### Parameters

<i>argc</i>	argument count
<i>argv</i>	array of strings

##### Returns

return INCORRECT\_PORT if port number is incorrect, INCORRECT\_IP if ip address is incorrect, INCORRECT\_ARG\_NUM if argument number is incorrect, CLIENT\_CONNECT\_ERROR if [client\\_connect\(\)](#) return error, else return 0

#### 2.1.4.2 client\_connect()

```
int client_connect (
    const int ip_addr,
    const int port )
```

function that use TCP protocol to connect to network with passed ip and port

##### Parameters

<i>ip_addr</i>	ip address to connect
<i>port</i>	port number

##### Returns

return CLIENT\_CONNECT\_ERROR if socket() or sendto() return error value

#### 2.1.4.3 server\_cli()

```
int server_cli (
    int argc,
    char * argv[] )
```

function that handle command line arguments and start server by using [server\\_start\(\)](#) if argument is correct

## Parameters

<i>argc</i>	argument count
<i>argv</i>	array of strings

## Returns

return INCORRECT\_PORT if port number is incorrect, INCORRECT\_ARG\_NUM if argument number is incorrect, return SERVER\_ERROR if [server\\_start\(\)](#) return error, else return 0

## 2.1.4.4 server\_start()

```
int server_start (
    int port )
```

start listen on chosen port through TCP protocol

## Parameters

<i>port</i>	chosen port
-------------	-------------

## Returns

SERVER\_ERROR if a server socket creating error or bind function error has occurred

## 2.2 tcp.h

[Go to the documentation of this file.](#)

```
1
2
3 #if !defined TCP_H_INCLUDED
4 #define TCP_H_INCLUDED
5
6 #include <stdio.h>
7
8
9 #define MAX_BUFFER_SIZE 1024
10 #define MAX_UNSIGNED_SHORT_VAL 65535
11 #define DEFAULT_PROTOCOL 0
12 #define SEND_FLAGS 0
13 #define RECV_FLAGS 0
14 #define DEFAULT_PORT 80
15 #define DEFAULT_IP INADDR_LOOPBACK
16
17 enum CLIENT_ERRORS {
18     INCORRECT_ARG_NUM = -5,
19     SERVER_ERROR = -4,
20     INCORRECT_PORT = -3,
21     INCORRECT_IP = -2,
22     CLIENT_CONNECT_ERROR = -1,
23     INCORRECT_SOCKET = -1,
24 };
25
26 enum CLI_ARGS_COUNT {
27     USR_CLI_DEFAULT_IP_N_PORT = 2,
28     USR_CLI_DEFAULT_PORT = 3,
29     USR_ALL_ARGS_PASSED = 4,
30     SERVER_CLI_DEFAULT_PORT = 2,
31     SERVER_ALL_ARGS_PASSED = 3,
32 };
```

```
53
54
55
62 int client_connect(const int ip_addr, const int port);
63
72 int client_cli(const int argc, char* argv[]);
73
78 int server_start(int port);
79
87 int server_cli(int argc, char* argv[]);
88
89
92 void write_help();
93
94 #endif
```

# Index

CLI\_ARGS\_COUNT  
tcp.h, 4

client\_cli  
tcp.h, 5

client\_connect  
tcp.h, 6

CLIENT\_CONNECT\_ERROR  
tcp.h, 5

CLIENT\_ERRORS  
tcp.h, 5

DEFAULT\_IP  
tcp.h, 4

DEFAULT\_PORT  
tcp.h, 4

DEFAULT\_PROTOCOL  
tcp.h, 4

INCORRECT\_ARG\_NUM  
tcp.h, 5

INCORRECT\_IP  
tcp.h, 5

INCORRECT\_PORT  
tcp.h, 5

INCORRECT\_SOCKET  
tcp.h, 5

MAX\_BUFFER\_SIZE  
tcp.h, 4

MAX\_UNSIGNED\_SHORT\_VAL  
tcp.h, 4

RECV\_FLAGS  
tcp.h, 4

SEND\_FLAGS  
tcp.h, 4

SERVER\_ALL\_ARGS\_PASSED  
tcp.h, 5

server\_cli  
tcp.h, 6

SERVER\_CLI\_DEFAULT\_PORT  
tcp.h, 5

SERVER\_ERROR  
tcp.h, 5

server\_start  
tcp.h, 7

tcp.h, 3

CLI\_ARGS\_COUNT, 4

client\_cli, 5

client\_connect, 6

CLIENT\_CONNECT\_ERROR, 5

CLIENT\_ERRORS, 5

DEFAULT\_IP, 4

DEFAULT\_PORT, 4

DEFAULT\_PROTOCOL, 4

INCORRECT\_ARG\_NUM, 5

INCORRECT\_IP, 5

INCORRECT\_PORT, 5

INCORRECT\_SOCKET, 5

MAX\_BUFFER\_SIZE, 4

MAX\_UNSIGNED\_SHORT\_VAL, 4

RECV\_FLAGS, 4

SEND\_FLAGS, 4

SERVER\_ALL\_ARGS\_PASSED, 5

server\_cli, 6

SERVER\_CLI\_DEFAULT\_PORT, 5

SERVER\_ERROR, 5

server\_start, 7

USR\_ALL\_ARGS\_PASSED, 5

USR\_CLI\_DEFAULT\_IP\_N\_PORT, 5

USR\_CLI\_DEFAULT\_PORT, 5

USR\_ALL\_ARGS\_PASSED  
tcp.h, 5

USR\_CLI\_DEFAULT\_IP\_N\_PORT  
tcp.h, 5

USR\_CLI\_DEFAULT\_PORT  
tcp.h, 5