

## My Awesome Server

Generated by Doxygen 1.8.17



<b>1 Data Structure Index</b>	<b>1</b>
1.1 Data Structures	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Data Structure Documentation</b>	<b>5</b>
3.1 http_request Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Field Documentation	5
3.1.2.1 headers	5
3.1.2.2 http_major	5
3.1.2.3 http_minor	6
3.1.2.4 method	6
3.1.2.5 target	6
3.2 server_config Struct Reference	6
3.2.1 Detailed Description	6
3.2.2 Field Documentation	6
3.2.2.1 listen_addr	7
3.2.2.2 mimes_file	7
3.2.2.3 port	7
3.2.2.4 website_root	7
3.3 web_stats Struct Reference	7
3.3.1 Detailed Description	7
3.3.2 Field Documentation	8
3.3.2.1 ko_400	8
3.3.2.2 ko_403	8
3.3.2.3 ko_404	8
3.3.2.4 ko_405	8
3.3.2.5 ok_200	8
3.3.2.6 served_connections	8
3.3.2.7 served_requests	8
<b>4 File Documentation</b>	<b>9</b>
4.1 webserver/config/config.c File Reference	9
4.1.1 Function Documentation	9
4.1.1.1 get_config()	10
4.1.1.2 get_config_from_file()	10
4.1.1.3 init_config()	10
4.1.2 Variable Documentation	10
4.1.2.1 shared_mem_config	11
4.2 webserver/config/config.h File Reference	11
4.2.1 Function Documentation	11

4.2.1.1	<a href="#">get_config()</a>	12
4.2.1.2	<a href="#">get_config_from_file()</a>	12
4.2.1.3	<a href="#">init_config()</a>	12
4.3	<a href="#">webserver/file.c File Reference</a>	12
4.3.1	<a href="#">Function Documentation</a>	13
4.3.1.1	<a href="#">check_and_open()</a>	13
4.3.1.2	<a href="#">check_root()</a>	14
4.3.1.3	<a href="#">copy()</a>	14
4.3.1.4	<a href="#">fgets_or_exit()</a>	14
4.3.1.5	<a href="#">get_app_path()</a>	15
4.3.1.6	<a href="#">get_file_size()</a>	15
4.3.1.7	<a href="#">get_mime_type()</a>	15
4.4	<a href="#">webserver/file.h File Reference</a>	16
4.4.1	<a href="#">Function Documentation</a>	17
4.4.1.1	<a href="#">check_and_open()</a>	17
4.4.1.2	<a href="#">check_root()</a>	17
4.4.1.3	<a href="#">copy()</a>	17
4.4.1.4	<a href="#">fgets_or_exit()</a>	18
4.4.1.5	<a href="#">get_app_path()</a>	18
4.4.1.6	<a href="#">get_file_size()</a>	18
4.4.1.7	<a href="#">get_mime_type()</a>	19
4.5	<a href="#">webserver/http/http.c File Reference</a>	19
4.5.1	<a href="#">Function Documentation</a>	20
4.5.1.1	<a href="#">get_date_http_format()</a>	20
4.5.1.2	<a href="#">rewrite_target()</a>	20
4.5.1.3	<a href="#">send_response()</a>	20
4.5.1.4	<a href="#">send_status()</a>	21
4.5.1.5	<a href="#">skip_headers()</a>	21
4.6	<a href="#">webserver/http/http.h File Reference</a>	22
4.6.1	<a href="#">Function Documentation</a>	22
4.6.1.1	<a href="#">get_date_http_format()</a>	22
4.6.1.2	<a href="#">rewrite_target()</a>	23
4.6.1.3	<a href="#">send_response()</a>	23
4.6.1.4	<a href="#">send_status()</a>	23
4.6.1.5	<a href="#">skip_headers()</a>	24
4.7	<a href="#">webserver/http/http_parse.c File Reference</a>	24
4.7.1	<a href="#">Macro Definition Documentation</a>	25
4.7.1.1	<a href="#">in_range</a>	25
4.7.1.2	<a href="#">min</a>	25
4.7.2	<a href="#">Function Documentation</a>	25
4.7.2.1	<a href="#">parse_http_request()</a>	25
4.8	<a href="#">webserver/http/http_parse.h File Reference</a>	26

4.8.1 Macro Definition Documentation	26
4.8.1.1 MAX_TARGET_SIZE	26
4.8.2 Enumeration Type Documentation	26
4.8.2.1 http_method	26
4.8.3 Function Documentation	27
4.8.3.1 parse_http_request()	27
4.9 webserver/log.c File Reference	27
4.9.1 Function Documentation	28
4.9.1.1 create_errors_logs_file()	28
4.9.1.2 create_requests_logs_file()	29
4.9.1.3 get_log_errors()	29
4.9.1.4 get_log_requests()	29
4.9.1.5 write_error()	29
4.9.1.6 write_request()	30
4.9.2 Variable Documentation	30
4.9.2.1 log_errors	30
4.9.2.2 log_requests	30
4.10 webserver/log.h File Reference	31
4.10.1 Function Documentation	31
4.10.1.1 create_errors_logs_file()	31
4.10.1.2 create_requests_logs_file()	32
4.10.1.3 get_log_errors()	32
4.10.1.4 get_log_requests()	32
4.10.1.5 write_error()	32
4.10.1.6 write_request()	33
4.10.2 Variable Documentation	33
4.10.2.1 client_ip	33
4.11 webserver/main.c File Reference	33
4.11.1 Function Documentation	34
4.11.1.1 child_handler()	34
4.11.1.2 init_signals()	34
4.11.1.3 main()	34
4.11.1.4 respond_client()	35
4.11.2 Variable Documentation	35
4.11.2.1 root	35
4.12 webserver/socket.c File Reference	35
4.12.1 Function Documentation	36
4.12.1.1 create_server()	36
4.13 webserver/socket.h File Reference	37
4.13.1 Function Documentation	37
4.13.1.1 create_server()	37
4.14 webserver/stats/stats.c File Reference	37

4.14.1 Function Documentation . . . . .	38
4.14.1.1 get_stats() . . . . .	38
4.14.1.2 init_stats() . . . . .	38
4.14.1.3 send_stats() . . . . .	38
4.14.2 Variable Documentation . . . . .	39
4.14.2.1 shared_memory . . . . .	39
4.15 webserver/stats/stats.h File Reference . . . . .	39
4.15.1 Function Documentation . . . . .	40
4.15.1.1 get_stats() . . . . .	40
4.15.1.2 init_stats() . . . . .	40
4.15.1.3 send_stats() . . . . .	40
4.15.2 Variable Documentation . . . . .	40
4.15.2.1 shared_semaphore . . . . .	40
<b>Index</b>	<b>41</b>

# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">http_request</a>	.....	5
<a href="#">server_config</a>	.....	6
<a href="#">web_stats</a>	.....	7





## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">webserver/file.c</a>	12
<a href="#">webserver/file.h</a>	16
<a href="#">webserver/log.c</a>	27
<a href="#">webserver/log.h</a>	31
<a href="#">webserver/main.c</a>	33
<a href="#">webserver/socket.c</a>	35
<a href="#">webserver/socket.h</a>	37
<a href="#">webserver/config/config.c</a>	9
<a href="#">webserver/config/config.h</a>	11
<a href="#">webserver/http/http.c</a>	19
<a href="#">webserver/http/http.h</a>	22
<a href="#">webserver/http/http_parse.c</a>	24
<a href="#">webserver/http/http_parse.h</a>	26
<a href="#">webserver/stats/stats.c</a>	37
<a href="#">webserver/stats/stats.h</a>	39



## Chapter 3

# Data Structure Documentation

### 3.1 http\_request Struct Reference

```
#include <http_parse.h>
```

#### Data Fields

- enum [http\\_method method](#)
- int [http\\_major](#)
- int [http\\_minor](#)
- char [target](#) [MAX\_TARGET\_SIZE]
- char \* [headers](#) [20]

#### 3.1.1 Detailed Description

describes a http request

#### 3.1.2 Field Documentation

##### 3.1.2.1 headers

```
char* http_request::headers[20]
```

headers of the request

##### 3.1.2.2 http\_major

```
int http_request::http_major
```

major HTTP version of the request

### 3.1.2.3 http\_minor

```
int http_request::http_minor
```

minor HTTP version of the request

### 3.1.2.4 method

```
enum http_method http_request::method
```

HTTP method of the request

### 3.1.2.5 target

```
char http_request::target[MAX_TARGET_SIZE]
```

target of the request

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

- [webserver/http/http\\_parse.h](#)

## 3.2 server\_config Struct Reference

```
#include <config.h>
```

### Data Fields

- int [port](#)
- char [listen\\_addr](#) [PATH\_MAX]
- char [website\\_root](#) [PATH\_MAX]
- char [mimes\\_file](#) [PATH\_MAX]

### 3.2.1 Detailed Description

struct for saving the configuration of the server

### 3.2.2 Field Documentation

### 3.2.2.1 listen\_addr

```
char server_config::listen_addr[PATH_MAX]
```

ip address to listen

### 3.2.2.2 mimes\_file

```
char server_config::mimes_file[PATH_MAX]
```

mime types file path

### 3.2.2.3 port

```
int server_config::port
```

port to listen

### 3.2.2.4 website\_root

```
char server_config::website_root[PATH_MAX]
```

web root of the site to serve

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

- [webserver/config/config.h](#)

## 3.3 web\_stats Struct Reference

```
#include <stats.h>
```

### Data Fields

- int [served\\_connections](#)
- int [served\\_requests](#)
- int [ok\\_200](#)
- int [ko\\_400](#)
- int [ko\\_403](#)
- int [ko\\_404](#)
- int [ko\\_405](#)

### 3.3.1 Detailed Description

struct for the stats

### 3.3.2 Field Documentation

#### 3.3.2.1 ko\_400

```
int web_stats::ko_400
```

number of 400 responses the server has sent

#### 3.3.2.2 ko\_403

```
int web_stats::ko_403
```

number of 403 responses the server has sent

#### 3.3.2.3 ko\_404

```
int web_stats::ko_404
```

number of 404 responses the server has sent

#### 3.3.2.4 ko\_405

```
int web_stats::ko_405
```

number of 405 responses the server has sent

#### 3.3.2.5 ok\_200

```
int web_stats::ok_200
```

number of 200 responses the server has sent

#### 3.3.2.6 served\_connections

```
int web_stats::served_connections
```

number of connections the server has received

#### 3.3.2.7 served\_requests

```
int web_stats::served_requests
```

number of request the server has received

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

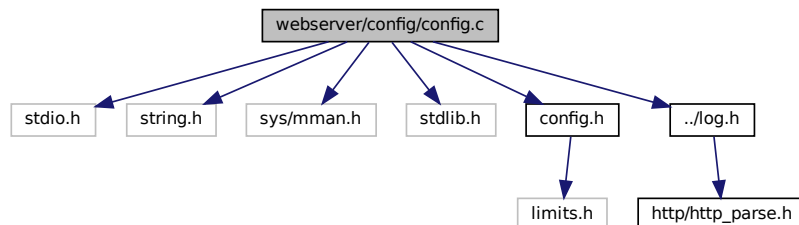
- [webserver/stats/stats.h](#)

## Chapter 4

# File Documentation

### 4.1 webserver/config/config.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <sys/mman.h>
#include <stdlib.h>
#include "config.h"
#include "../log.h"
Include dependency graph for config.c:
```



#### Functions

- `int init_config (char *abs_path)`
- `int get_config_from_file (char *abs_path)`
- `server_config * get_config (void)`

#### Variables

- `server_config * shared_mem_config`

#### 4.1.1 Function Documentation

#### 4.1.1.1 get\_config()

```
server_config* get_config (
    void )
```

return server config

##### Returns

a pointer to the shared memory zone with the configuration of the server inside

#### 4.1.1.2 get\_config\_from\_file()

```
int get_config_from_file (
    char * abs_path )
```

open the config file and read it

##### Parameters

<i>abs_path</i>	absolute path of the application
-----------------	----------------------------------

##### Returns

0 on success, 1 on error

#### 4.1.1.3 init\_config()

```
int init_config (
    char * abs_path )
```

Init the configuration of the server

##### Parameters

<i>abs_path</i>	absolute path of the app
-----------------	--------------------------

##### Returns

0 on success, 1 on error

### 4.1.2 Variable Documentation



#### 4.1.2.1 shared\_mem\_config

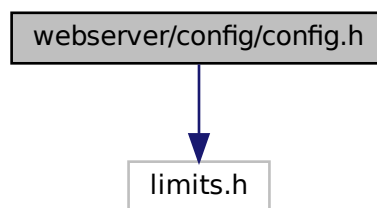
```
server_config* shared_mem_config
```

shared memory zone for access config in the whole application

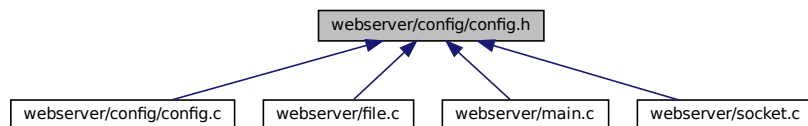
## 4.2 webserver/config/config.h File Reference

```
#include <limits.h>
```

Include dependency graph for config.h:



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



## Data Structures

- struct [server\\_config](#)

## Functions

- int [init\\_config](#) (char \*abs\_path)
- int [get\\_config\\_from\\_file](#) (char \*abs\_path)
- [server\\_config](#) \* [get\\_config](#) (void)

### 4.2.1 Function Documentation

#### 4.2.1.1 get\_config()

```
server_config* get_config (
    void )
```

return server config

##### Returns

a pointer to the shared memory zone with the configuration of the server inside

#### 4.2.1.2 get\_config\_from\_file()

```
int get_config_from_file (
    char * abs_path )
```

open the config file and read it

##### Parameters

<i>abs_path</i>	absolute path of the application
-----------------	----------------------------------

##### Returns

0 on success, 1 on error

#### 4.2.1.3 init\_config()

```
int init_config (
    char * abs_path )
```

Init the configuration of the server

##### Parameters

<i>abs_path</i>	absolute path of the app
-----------------	--------------------------

##### Returns

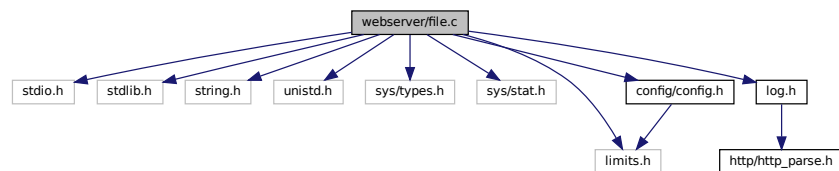
0 on success, 1 on error

## 4.3 webserver/file.c File Reference

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

```
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <limits.h>
#include "config/config.h"
#include "log.h"
```

Include dependency graph for file.c:



## Functions

- FILE \* [check\\_and\\_open](#) (const char \*target, const char \*document\_root)
- int [get\\_file\\_size](#) (int fd)
- void [copy](#) (FILE \*in, FILE \*out)
- char \* [fgets\\_or\\_exit](#) (char \*buffer, int size, FILE \*stream)
- char \* [check\\_root](#) (char \*root)
- char \* [get\\_mime\\_type](#) (char \*name)
- char \* [get\\_app\\_path](#) (char \*argv0)

### 4.3.1 Function Documentation

#### 4.3.1.1 check\_and\_open()

```
FILE* check_and_open (
    const char * target,
    const char * document_root )
```

function to check if the file of the target exist, check if we can open it and open it

##### Parameters

<i>target</i>	the target of the request
<i>document_root</i>	the root path of the website

##### Returns

a pointer to the opened file

#### 4.3.1.2 check\_root()

```
char* check_root (
    char * root )
```

check if we can open the root of the website

##### Parameters

<i>root</i>	the path to the root
-------------	----------------------

##### Returns

the root after the check

#### 4.3.1.3 copy()

```
void copy (
    FILE * in,
    FILE * out )
```

copy the content of the file to another

##### Parameters

<i>in</i>	the file to read
<i>out</i>	the file to copy data

#### 4.3.1.4 fgets\_or\_exit()

```
char* fgets_or_exit (
    char * buffer,
    int size,
    FILE * stream )
```

read data and if it fail quit the program with error code

##### Parameters

<i>buffer</i>	buffer to store data
<i>size</i>	the size of data we read
<i>stream</i>	the stream to read data

**Returns**

the buffer

**4.3.1.5 get\_app\_path()**

```
char* get_app_path (
    char * argv0 )
```

return the path of the application

**Parameters**

<i>argv0</i>	the path of the executable
--------------	----------------------------

**Returns**

absolute path of the file

**4.3.1.6 get\_file\_size()**

```
int get_file_size (
    int fd )
```

find the size of a file

**Parameters**

<i>fd</i>	the file descriptor for the file to open
-----------	--

**Returns**

the size of the file

**4.3.1.7 get\_mime\_type()**

```
char* get_mime_type (
    char * name )
```

return the mime type of a file

**Parameters**

<i>name</i>	the name of the file
-------------	----------------------

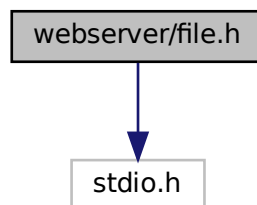
**Returns**

the mime type of the file

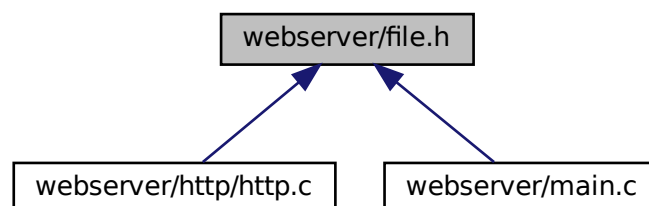
## 4.4 webserver/file.h File Reference

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

Include dependency graph for file.h:



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

**Functions**

- FILE \* [check\\_and\\_open](#) (const char \*target, const char \*document\_root)
- int [get\\_file\\_size](#) (int fd)
- void [copy](#) (FILE \*in, FILE \*out)
- char \* [fgets\\_or\\_exit](#) (char \*buffer, int size, FILE \*stream)
- char \* [check\\_root](#) (char \*root)
- char \* [get\\_mime\\_type](#) (char \*name)
- char \* [get\\_app\\_path](#) (char \*argv0)

## 4.4.1 Function Documentation

### 4.4.1.1 `check_and_open()`

```
FILE* check_and_open (
    const char * target,
    const char * document_root )
```

function to check if the file of the target exist, check if we can open it and open it

#### Parameters

<i>target</i>	the target of the request
<i>document_root</i>	the root path of the website

#### Returns

a pointer to the opened file

### 4.4.1.2 `check_root()`

```
char* check_root (
    char * root )
```

check if we can open the root of the website

#### Parameters

<i>root</i>	the path to the root
-------------	----------------------

#### Returns

the root after the check

### 4.4.1.3 `copy()`

```
void copy (
    FILE * in,
    FILE * out )
```

copy the content of the file to another

**Parameters**

<i>in</i>	the file to read
<i>out</i>	the file to copy data

**4.4.1.4 fgets\_or\_exit()**

```
char* fgets_or_exit (
    char * buffer,
    int size,
    FILE * stream )
```

read data and if it fail quit the program with error code

**Parameters**

<i>buffer</i>	buffer to store data
<i>size</i>	the size of data we read
<i>stream</i>	the stream to read data

**Returns**

the buffer

**4.4.1.5 get\_app\_path()**

```
char* get_app_path (
    char * argv0 )
```

return the path of the application

**Parameters**

<i>argv0</i>	the path of the executable
--------------	----------------------------

**Returns**

absolute path of the file

**4.4.1.6 get\_file\_size()**

```
int get_file_size (
    int fd )
```



find the size of a file

#### Parameters

<i>fd</i>	the file descriptor for the file to open
-----------	--

#### Returns

the size of the file

#### 4.4.1.7 get\_mime\_type()

```
char* get_mime_type (  
    char * name )
```

return the mime type of a file

#### Parameters

<i>name</i>	the name of the file
-------------	----------------------

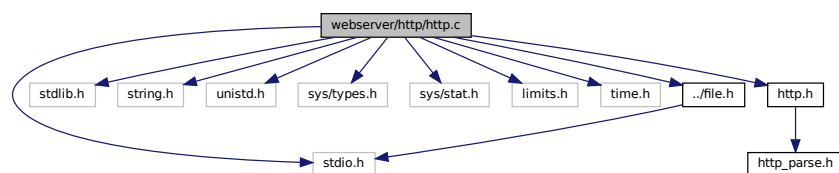
#### Returns

the mime type of the file

## 4.5 webserver/http/http.c File Reference

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <unistd.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <limits.h>  
#include <time.h>  
#include "http.h"  
#include "../file.h"
```

Include dependency graph for http.c:



## Functions

- void [skip\\_headers](#) (FILE \*client, [http\\_request](#) \*request)
- void [send\\_status](#) (FILE \*client, int code, const char \*reason\_phrase)
- void [send\\_response](#) (FILE \*client, int code, const char \*reason\_phrase, char \*message\_body, int size)
- char \* [get\\_date\\_http\\_format](#) (void)
- char \* [rewrite\\_target](#) (char \*target)

### 4.5.1 Function Documentation

#### 4.5.1.1 [get\\_date\\_http\\_format\(\)](#)

```
char* get_date_http_format (  
    void )
```

return actual date of the server with the correct format for HTTP response

##### Returns

well formatted date

#### 4.5.1.2 [rewrite\\_target\(\)](#)

```
char* rewrite_target (  
    char * target )
```

rewrite the HTTP target within URLs variables

##### Parameters

<i>target</i>	the target of the request
---------------	---------------------------

##### Returns

well rewrite target

#### 4.5.1.3 [send\\_response\(\)](#)

```
void send_response (  
    FILE * client,  
    int code,
```

```
const char * reason_phrase,  
char * message_body,  
int size )
```

function to format the HTTP response

#### Parameters

<i>client</i>	stream to write the response
<i>code</i>	the HTTP code of the response
<i>reason_phrase</i>	the reason phrase of the response
<i>message_body</i>	the body of the response
<i>size</i>	the size of the response body

#### 4.5.1.4 send\_status()

```
void send_status (  
    FILE * client,  
    int code,  
    const char * reason_phrase )
```

send the status of the response

#### Parameters

<i>client</i>	stream to send data
<i>code</i>	HTTP code of the response
<i>reason_phrase</i>	HTTP response reason phrase

#### 4.5.1.5 skip\_headers()

```
void skip_headers (  
    FILE * client,  
    http_request * request )
```

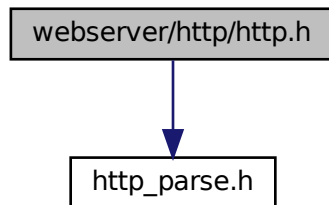
ignore headers of yhe requests

#### Parameters

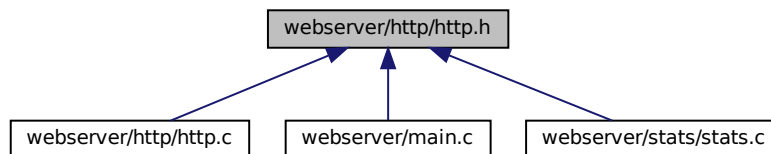
<i>client</i>	request stream
<i>request</i>	request we parse

## 4.6 webservice/http/http.h File Reference

```
#include "http_parse.h"
Include dependency graph for http.h:
```



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



### Functions

- void [skip\\_headers](#) (FILE \*client, [http\\_request](#) \*request)
- void [send\\_status](#) (FILE \*client, int code, const char \*reason\_phrase)
- void [send\\_response](#) (FILE \*client, int code, const char \*reason\_phrase, char \*message\_body, int size)
- char \* [get\\_date\\_http\\_format](#) (void)
- char \* [rewrite\\_target](#) (char \*target)

### 4.6.1 Function Documentation

#### 4.6.1.1 [get\\_date\\_http\\_format\(\)](#)

```
char* get_date_http_format (
    void )
```

return actual date of the server with the correct format for HTTP response

#### Returns

well formatted date

#### 4.6.1.2 `rewrite_target()`

```
char* rewrite_target (
    char * target )
```

rewrite the HTTP target within URLs variables

##### Parameters

<i>target</i>	the target of the request
---------------	---------------------------

##### Returns

well rewrite target

#### 4.6.1.3 `send_response()`

```
void send_response (
    FILE * client,
    int code,
    const char * reason_phrase,
    char * message_body,
    int size )
```

function to format the HTTP response

##### Parameters

<i>client</i>	stream to write the response
<i>code</i>	the HTTP code of the response
<i>reason_phrase</i>	the reason phrase of the response
<i>message_body</i>	the body of the response
<i>size</i>	the size of the response body

#### 4.6.1.4 `send_status()`

```
void send_status (
    FILE * client,
    int code,
    const char * reason_phrase )
```

send the status of the response

##### Parameters

<i>client</i>	stream to send data
<i>code</i>	HTTP code of the response
<i>reason_phrase</i>	HTTP response reason phrase

#### 4.6.1.5 skip\_headers()

```
void skip_headers (
    FILE * client,
    http_request * request )
```

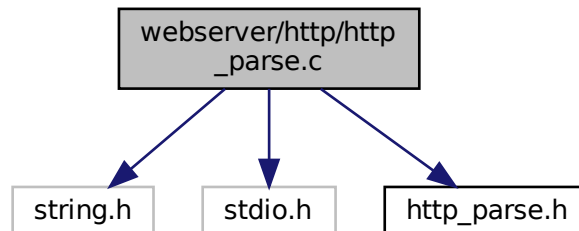
ignore headers of yhe requests

##### Parameters

<i>client</i>	request stream
<i>request</i>	request we parse

## 4.7 webserver/http/http\_parse.c File Reference

```
#include <string.h>
#include <stdio.h>
#include "http_parse.h"
Include dependency graph for http_parse.c:
```



## Macros

- `#define min(a, b) ((a) < (b) ? (a) : (b))`
- `#define in_range(a, b, c) ((a) < (b) ? 0 : ((a) > (c) ? 0 : 1))`

## Functions

- `int parse_http_request (const char *request_line, http_request *request)`

## 4.7.1 Macro Definition Documentation

### 4.7.1.1 in\_range

```
#define in_range(  
    a,  
    b,  
    c ) ((a) < (b) ? 0 : ((a) > (c) ? 0 : 1))
```

find if a number is in two others

### 4.7.1.2 min

```
#define min(  
    a,  
    b ) ((a) < (b) ? (a) : (b))
```

find the minimum value between two numbers

## 4.7.2 Function Documentation

### 4.7.2.1 parse\_http\_request()

```
int parse_http_request (  
    const char * request_line,  
    http_request * request )
```

function to parse the request

#### Parameters

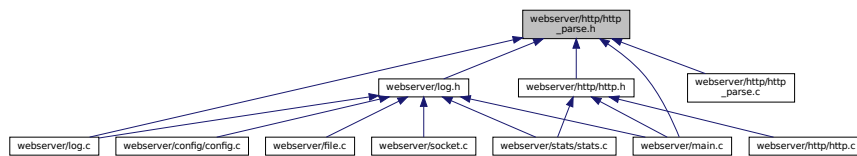
<i>request_line</i>	main request line
<i>request</i>	request to parse

**Returns**

1 on success, 0 on error

**4.8 webserver/http/http\_parse.h File Reference**

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

**Data Structures**

- struct [http\\_request](#)

**Macros**

- #define [MAX\\_TARGET\\_SIZE](#) 1024

**Enumerations**

- enum [http\\_method](#) { [HTTP\\_GET](#), [HTTP\\_HEAD](#), [HTTP\\_UNSUPPORTED](#) }

**Functions**

- int [parse\\_http\\_request](#) (const char \*request\_line, [http\\_request](#) \*request)

**4.8.1 Macro Definition Documentation****4.8.1.1 MAX\_TARGET\_SIZE**

```
#define MAX_TARGET_SIZE 1024
```

**4.8.2 Enumeration Type Documentation****4.8.2.1 http\_method**

```
enum http\_method
```

Method supported by the parser



## Enumerator

HTTP_GET	GET http method
HTTP_HEAD	HEAD http method
HTTP_UNSUPPORTED	the method is not supported

### 4.8.3 Function Documentation

#### 4.8.3.1 parse\_http\_request()

```
int parse_http_request (
    const char * request_line,
    http_request * request )
```

Parses a http request line.

## Parameters

<i>in</i>	request_line the line to parse (as a null terminated string)
<i>out</i>	request a valid pointer to a <a href="#">http_request</a> that will be filled by the function

## Returns

-1 if error and 0 on success. If the error is an unsupported http method, then the method field of request will be set to HTTP\_UNSUPPORTED

function to parse the request

## Parameters

<i>request_line</i>	main request line
<i>request</i>	request to parse

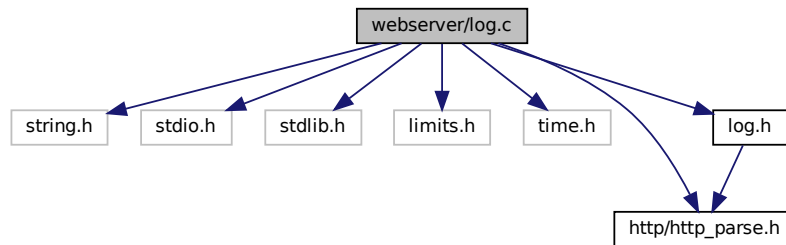
## Returns

1 on success, 0 on error

## 4.9 webserver/log.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <time.h>
```

```
#include "http/http_parse.h"
#include "log.h"
Include dependency graph for log.c:
```



## Functions

- void [create\\_requests\\_logs\\_file](#) (char \*path)
- void [create\\_errors\\_logs\\_file](#) (char \*path)
- void [write\\_request](#) (FILE \*log\_file, [http\\_request](#) request, int code)
- void [write\\_error](#) (FILE \*log\_file, char \*error)
- FILE \* [get\\_log\\_requests](#) (void)
- FILE \* [get\\_log\\_errors](#) (void)

## Variables

- FILE \* [log\\_requests](#)
- FILE \* [log\\_errors](#)

## 4.9.1 Function Documentation

### 4.9.1.1 [create\\_errors\\_logs\\_file\(\)](#)

```
void create_errors_logs_file (
    char * path )
```

init the errors log file

#### Parameters

<i>path</i>	the path to the log directory
-------------	-------------------------------

#### 4.9.1.2 create\_requests\_logs\_file()

```
void create_requests_logs_file (
    char * path )
```

init the requests log file

##### Parameters

<i>path</i>	the path to the log directory
-------------	-------------------------------

#### 4.9.1.3 get\_log\_errors()

```
FILE* get_log_errors (
    void )
```

return a file descriptor to the error log file

##### Returns

the file descriptor

#### 4.9.1.4 get\_log\_requests()

```
FILE* get_log_requests (
    void )
```

return a file descriptor to the requests log file

##### Returns

the file descriptor

#### 4.9.1.5 write\_error()

```
void write_error (
    FILE * log_file,
    char * error )
```

function to write an error in the log file

**Parameters**

<i>log_file</i>	the log file
<i>request</i>	the error to write

**4.9.1.6 write\_request()**

```
void write_request (
    FILE * log_file,
    http_request request,
    int code )
```

function to write a request in the log file

**Parameters**

<i>log_file</i>	the log file
<i>request</i>	the request to write
<i>code</i>	the HTTP code of the request

**4.9.2 Variable Documentation****4.9.2.1 log\_errors**

```
FILE* log_errors
```

the log file for errors

**4.9.2.2 log\_requests**

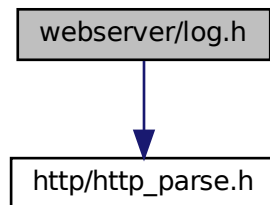
```
FILE* log_requests
```

the log file for requests

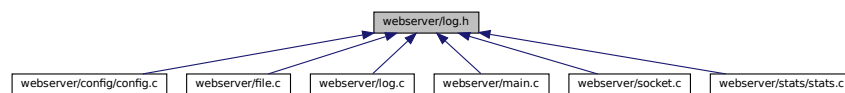
## 4.10 webserver/log.h File Reference

```
#include "http/http_parse.h"
```

Include dependency graph for log.h:



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



### Functions

- void [create\\_requests\\_logs\\_file](#) (char \*path)
- void [create\\_errors\\_logs\\_file](#) (char \*path)
- void [write\\_request](#) (FILE \*log\_file, [http\\_request](#) request, int code)
- void [write\\_error](#) (FILE \*log\_file, char \*error)
- FILE \* [get\\_log\\_requests](#) (void)
- FILE \* [get\\_log\\_errors](#) (void)

### Variables

- char [client\\_ip](#) [20]

#### 4.10.1 Function Documentation

##### 4.10.1.1 create\_errors\_logs\_file()

```
void create_errors_logs_file (
    char * path )
```

init the errors log file

**Parameters**

<i>path</i>	the path to the log directory
-------------	-------------------------------

**4.10.1.2 create\_requests\_logs\_file()**

```
void create_requests_logs_file (  
    char * path )
```

init the requests log file

**Parameters**

<i>path</i>	the path to the log directory
-------------	-------------------------------

**4.10.1.3 get\_log\_errors()**

```
FILE* get_log_errors (  
    void )
```

return a file descriptor to the error log file

**Returns**

the file descriptor

**4.10.1.4 get\_log\_requests()**

```
FILE* get_log_requests (  
    void )
```

return a file descriptor to the requests log file

**Returns**

the file descriptor

**4.10.1.5 write\_error()**

```
void write_error (  
    FILE * log_file,  
    char * error )
```

function to write an error in the log file

## Parameters

<i>log_file</i>	the log file
<i>request</i>	the error to write

**4.10.1.6 write\_request()**

```
void write_request (
    FILE * log_file,
    http_request request,
    int code )
```

function to write a request in the log file

## Parameters

<i>log_file</i>	the log file
<i>request</i>	the request to write
<i>code</i>	the HTTP code of the request

**4.10.2 Variable Documentation****4.10.2.1 client\_ip**

```
char client_ip[20]
```

the ip address of the client

**4.11 webserver/main.c File Reference**

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <netinet/in.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <limits.h>
#include <semaphore.h>
#include <arpa/inet.h>
#include "socket.h"
```

```
#include "http/http_parse.h"
#include "stats/stats.h"
#include "config/config.h"
#include "http/http.h"
#include "file.h"
#include "log.h"
```

Include dependency graph for main.c:



## Functions

- void [init\\_signals](#) (void)
- void [respond\\_client](#) (int socket\_client)
- void [child\\_handler](#) (void)
- int [main](#) (int argc, char \*argv[ ])

## Variables

- char [root](#) [PATH\_MAX]

### 4.11.1 Function Documentation

#### 4.11.1.1 child\_handler()

```
void child_handler (
    void )
```

function to dismiss zombies process

#### 4.11.1.2 init\_signals()

```
void init_signals (
    void )
```

function to ignore SIGPIPE signal

#### 4.11.1.3 main()

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

the main function of the program



**Parameters**

<i>argc</i>	arguments counter
<i>argv</i>	arguments array

**Returns**

0 on success, positive value on error

**4.11.1.4 respond\_client()**

```
void respond_client (
    int socket_client )
```

function to respond to the client

**Parameters**

<i>socket_client</i>	the socket client
----------------------	-------------------

**4.11.2 Variable Documentation****4.11.2.1 root**

```
char root[PATH_MAX]
```

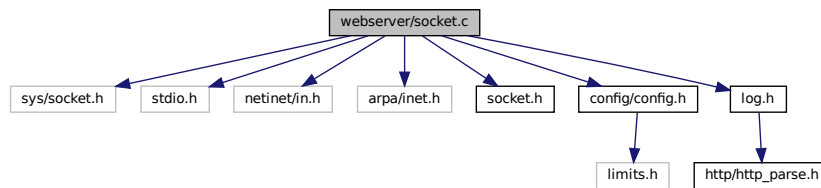
the root directory of the application

**4.12 webserver/socket.c File Reference**

```
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include "socket.h"
#include "config/config.h"
```

```
#include "log.h"
```

Include dependency graph for socket.c:



## Functions

- int [create\\_server](#) (int port)

### 4.12.1 Function Documentation

#### 4.12.1.1 create\_server()

```
int create_server (  
    int port )
```

creation of the server with the port we want to listen

##### Parameters

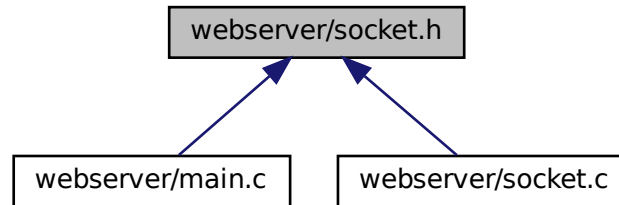
<i>port</i>	the port to listen
-------------	--------------------

##### Returns

the server socket

## 4.13 webserver/socket.h File Reference

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



### Functions

- int [create\\_server](#) (int port)

#### 4.13.1 Function Documentation

##### 4.13.1.1 create\_server()

```
int create_server (  
    int port )
```

creation of the server with the port we want to listen

#### Parameters

<i>port</i>	the port to listen
-------------	--------------------

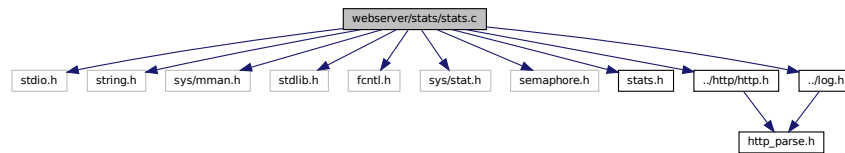
#### Returns

the server socket

## 4.14 webserver/stats/stats.c File Reference

```
#include <stdio.h>  
#include <string.h>  
#include <sys/mman.h>  
#include <stdlib.h>
```

```
#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>
#include "stats.h"
#include "../http/http.h"
#include "../log.h"
Include dependency graph for stats.c:
```



## Functions

- void [send\\_stats](#) (FILE \*client)
- void [init\\_stats](#) (void)
- [web\\_stats](#) \* [get\\_stats](#) (void)

## Variables

- [web\\_stats](#) \* [shared\\_memory](#)

## 4.14.1 Function Documentation

### 4.14.1.1 [get\\_stats\(\)](#)

```
web\_stats* get\_stats (
    void )
```

return the shared memory zone of the stats

#### Returns

a pointer to the server stats

### 4.14.1.2 [init\\_stats\(\)](#)

```
void init\_stats (
    void )
```

init the server stats, the semaphore and the shared memory

### 4.14.1.3 [send\\_stats\(\)](#)

```
void send\_stats (
    FILE * client )
```

function who display stats of the server in html

## Parameters

<i>client</i>	the socket of the client
---------------	--------------------------

## 4.14.2 Variable Documentation

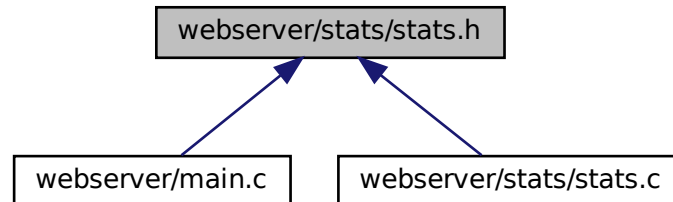
### 4.14.2.1 shared\_memory

`web_stats*` `shared_memory`

shared memory for the stats

## 4.15 webserver/stats/stats.h File Reference

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



## Data Structures

- struct `web_stats`

## Functions

- void `send_stats` (FILE \*`client`)
- void `init_stats` (void)
- `web_stats *` `get_stats` (void)

## Variables

- `sem_t *` `shared_semaphore`

## 4.15.1 Function Documentation

### 4.15.1.1 `get_stats()`

```
web_stats* get_stats (
    void )
```

return the shared memory zone of the stats

#### Returns

a pointer to the server stats

### 4.15.1.2 `init_stats()`

```
void init_stats (
    void )
```

init the server stats, the semaphore and the shared memory

### 4.15.1.3 `send_stats()`

```
void send_stats (
    FILE * client )
```

function who display stats of the server in html

#### Parameters

<i>client</i>	the socket of the client
---------------	--------------------------

## 4.15.2 Variable Documentation

### 4.15.2.1 `shared_semaphore`

```
sem_t* shared_semaphore
```

semaphore to avoid concurrent access to the stats

# Index

- check\_and\_open
  - file.c, [13](#)
  - file.h, [17](#)
- check\_root
  - file.c, [13](#)
  - file.h, [17](#)
- child\_handler
  - main.c, [34](#)
- client\_ip
  - log.h, [33](#)
- config.c
  - get\_config, [9](#)
  - get\_config\_from\_file, [10](#)
  - init\_config, [10](#)
  - shared\_mem\_config, [10](#)
- config.h
  - get\_config, [11](#)
  - get\_config\_from\_file, [12](#)
  - init\_config, [12](#)
- copy
  - file.c, [14](#)
  - file.h, [17](#)
- create\_errors\_logs\_file
  - log.c, [28](#)
  - log.h, [31](#)
- create\_requests\_logs\_file
  - log.c, [28](#)
  - log.h, [32](#)
- create\_server
  - socket.c, [36](#)
  - socket.h, [37](#)
- fgets\_or\_exit
  - file.c, [14](#)
  - file.h, [18](#)
- file.c
  - check\_and\_open, [13](#)
  - check\_root, [13](#)
  - copy, [14](#)
  - fgets\_or\_exit, [14](#)
  - get\_app\_path, [15](#)
  - get\_file\_size, [15](#)
  - get\_mime\_type, [15](#)
- file.h
  - check\_and\_open, [17](#)
  - check\_root, [17](#)
  - copy, [17](#)
  - fgets\_or\_exit, [18](#)
  - get\_app\_path, [18](#)
  - get\_file\_size, [18](#)
  - get\_mime\_type, [19](#)
- get\_app\_path
  - file.c, [15](#)
  - file.h, [18](#)
- get\_config
  - config.c, [9](#)
  - config.h, [11](#)
- get\_config\_from\_file
  - config.c, [10](#)
  - config.h, [12](#)
- get\_date\_http\_format
  - http.c, [20](#)
  - http.h, [22](#)
- get\_file\_size
  - file.c, [15](#)
  - file.h, [18](#)
- get\_log\_errors
  - log.c, [29](#)
  - log.h, [32](#)
- get\_log\_requests
  - log.c, [29](#)
  - log.h, [32](#)
- get\_mime\_type
  - file.c, [15](#)
  - file.h, [19](#)
- get\_stats
  - stats.c, [38](#)
  - stats.h, [40](#)
- headers
  - http\_request, [5](#)
- http.c
  - get\_date\_http\_format, [20](#)
  - rewrite\_target, [20](#)
  - send\_response, [20](#)
  - send\_status, [21](#)
  - skip\_headers, [21](#)
- http.h
  - get\_date\_http\_format, [22](#)
  - rewrite\_target, [22](#)
  - send\_response, [23](#)
  - send\_status, [23](#)
  - skip\_headers, [24](#)
- HTTP\_GET
  - http\_parse.h, [27](#)
- HTTP\_HEAD
  - http\_parse.h, [27](#)
- http\_major
  - http\_request, [5](#)

- http\_method
  - http\_parse.h, 26
- http\_minor
  - http\_request, 5
- http\_parse.c
  - in\_range, 25
  - min, 25
  - parse\_http\_request, 25
- http\_parse.h
  - HTTP\_GET, 27
  - HTTP\_HEAD, 27
  - http\_method, 26
  - HTTP\_UNSUPPORTED, 27
  - MAX\_TARGET\_SIZE, 26
  - parse\_http\_request, 27
- http\_request, 5
  - headers, 5
  - http\_major, 5
  - http\_minor, 5
  - method, 6
  - target, 6
- HTTP\_UNSUPPORTED
  - http\_parse.h, 27
- in\_range
  - http\_parse.c, 25
- init\_config
  - config.c, 10
  - config.h, 12
- init\_signals
  - main.c, 34
- init\_stats
  - stats.c, 38
  - stats.h, 40
- ko\_400
  - web\_stats, 8
- ko\_403
  - web\_stats, 8
- ko\_404
  - web\_stats, 8
- ko\_405
  - web\_stats, 8
- listen\_addr
  - server\_config, 6
- log.c
  - create\_errors\_logs\_file, 28
  - create\_requests\_logs\_file, 28
  - get\_log\_errors, 29
  - get\_log\_requests, 29
  - log\_errors, 30
  - log\_requests, 30
  - write\_error, 29
  - write\_request, 30
- log.h
  - client\_ip, 33
  - create\_errors\_logs\_file, 31
  - create\_requests\_logs\_file, 32
  - get\_log\_errors, 32
  - get\_log\_requests, 32
  - write\_error, 32
  - write\_request, 33
- log\_errors
  - log.c, 30
- log\_requests
  - log.c, 30
- main
  - main.c, 34
- main.c
  - child\_handler, 34
  - init\_signals, 34
  - main, 34
  - respond\_client, 35
  - root, 35
- MAX\_TARGET\_SIZE
  - http\_parse.h, 26
- method
  - http\_request, 6
- mimes\_file
  - server\_config, 7
- min
  - http\_parse.c, 25
- ok\_200
  - web\_stats, 8
- parse\_http\_request
  - http\_parse.c, 25
  - http\_parse.h, 27
- port
  - server\_config, 7
- respond\_client
  - main.c, 35
- rewrite\_target
  - http.c, 20
  - http.h, 22
- root
  - main.c, 35
- send\_response
  - http.c, 20
  - http.h, 23
- send\_stats
  - stats.c, 38
  - stats.h, 40
- send\_status
  - http.c, 21
  - http.h, 23
- served\_connections
  - web\_stats, 8
- served\_requests
  - web\_stats, 8
- server\_config, 6
  - listen\_addr, 6
  - mimes\_file, 7



- port, [7](#)
- website\_root, [7](#)
- shared\_mem\_config
  - config.c, [10](#)
- shared\_memory
  - stats.c, [39](#)
- shared\_semaphore
  - stats.h, [40](#)
- skip\_headers
  - http.c, [21](#)
  - http.h, [24](#)
- socket.c
  - create\_server, [36](#)
- socket.h
  - create\_server, [37](#)
- stats.c
  - get\_stats, [38](#)
  - init\_stats, [38](#)
  - send\_stats, [38](#)
  - shared\_memory, [39](#)
- stats.h
  - get\_stats, [40](#)
  - init\_stats, [40](#)
  - send\_stats, [40](#)
  - shared\_semaphore, [40](#)
- target
  - http\_request, [6](#)
- web\_stats, [7](#)
  - ko\_400, [8](#)
  - ko\_403, [8](#)
  - ko\_404, [8](#)
  - ko\_405, [8](#)
  - ok\_200, [8](#)
  - served\_connections, [8](#)
  - served\_requests, [8](#)
- webserver/config/config.c, [9](#)
- webserver/config/config.h, [11](#)
- webserver/file.c, [12](#)
- webserver/file.h, [16](#)
- webserver/http/http.c, [19](#)
- webserver/http/http.h, [22](#)
- webserver/http/http\_parse.c, [24](#)
- webserver/http/http\_parse.h, [26](#)
- webserver/log.c, [27](#)
- webserver/log.h, [31](#)
- webserver/main.c, [33](#)
- webserver/socket.c, [35](#)
- webserver/socket.h, [37](#)
- webserver/stats/stats.c, [37](#)
- webserver/stats/stats.h, [39](#)
- website\_root
  - server\_config, [7](#)
- write\_error
  - log.c, [29](#)
  - log.h, [32](#)
- write\_request
  - log.c, [30](#)
- log.h, [33](#)