

My Awesome Server

Generated by Doxygen 1.8.17

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
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	6
3.2.2.2 mimes_file	7
3.2.2.3 port	7
3.3 web_stats Struct Reference	7
3.3.1 Detailed Description	7
3.3.2 Field Documentation	7
3.3.2.1 ko_400	7
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
4 File Documentation	9
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 get_config()	12

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

4.8 webserver/http/http_parse.h File Reference	26
4.8.1 Macro Definition Documentation	27
4.8.1.1 MAX_TARGET_SIZE	27
4.8.2 Enumeration Type Documentation	27
4.8.2.1 http_method	27
4.8.3 Function Documentation	28
4.8.3.1 init_request()	28
4.8.3.2 parse_http_request()	28
4.9 webserver/log.c File Reference	28
4.9.1 Function Documentation	29
4.9.1.1 create_errors_logs_file()	29
4.9.1.2 create_requests_logs_file()	30
4.9.1.3 get_log_errors()	30
4.9.1.4 get_log_requests()	30
4.9.1.5 write_error()	30
4.9.1.6 write_request()	31
4.9.2 Variable Documentation	31
4.9.2.1 log_errors	31
4.9.2.2 log_requests	31
4.10 webserver/log.h File Reference	32
4.10.1 Function Documentation	32
4.10.1.1 create_errors_logs_file()	32
4.10.1.2 create_requests_logs_file()	33
4.10.1.3 get_log_errors()	33
4.10.1.4 get_log_requests()	33
4.10.1.5 write_error()	33
4.10.1.6 write_request()	34
4.10.2 Variable Documentation	34
4.10.2.1 client_ip	34
4.11 webserver/main.c File Reference	34
4.11.1 Function Documentation	35
4.11.1.1 child_handler()	35
4.11.1.2 init_signals()	35
4.11.1.3 main()	35
4.11.1.4 respond_client()	36
4.11.2 Variable Documentation	36
4.11.2.1 root	36
4.12 webserver/socket.c File Reference	36
4.12.1 Function Documentation	37
4.12.1.1 create_server()	37
4.13 webserver/socket.h File Reference	38
4.13.1 Function Documentation	38

4.13.1.1 create_server()	38
4.14 webserver/stats/stats.c File Reference	38
4.14.1 Function Documentation	39
4.14.1.1 get_stats()	39
4.14.1.2 init_stats()	39
4.14.1.3 send_stats()	39
4.14.2 Variable Documentation	40
4.14.2.1 shared_memory	40
4.15 webserver/stats/stats.h File Reference	40
4.15.1 Function Documentation	41
4.15.1.1 get_stats()	41
4.15.1.2 init_stats()	41
4.15.1.3 send_stats()	41
4.15.2 Variable Documentation	41
4.15.2.1 shared_semaphore	41
4.16 webserver/vhosts/hosts.c File Reference	42
4.16.1 Function Documentation	42
4.16.1.1 get_vhost_root()	42
4.16.1.2 read_host_file()	43
4.17 webserver/vhosts/hosts.h File Reference	43
4.17.1 Function Documentation	44
4.17.1.1 get_vhost_root()	44
Index	45

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

http_request	5
server_config	6
web_stats	7

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

webserver/ file.c	12
webserver/ file.h	16
webserver/ log.c	28
webserver/ log.h	32
webserver/ main.c	34
webserver/ socket.c	36
webserver/ socket.h	38
webserver/config/ config.c	9
webserver/config/ config.h	11
webserver/http/ http.c	19
webserver/http/ http.h	22
webserver/http/ http_parse.c	25
webserver/http/ http_parse.h	26
webserver/stats/ stats.c	38
webserver/stats/ stats.h	40
webserver/vhosts/ hosts.c	42
webserver/vhosts/ hosts.h	43

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 [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

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:

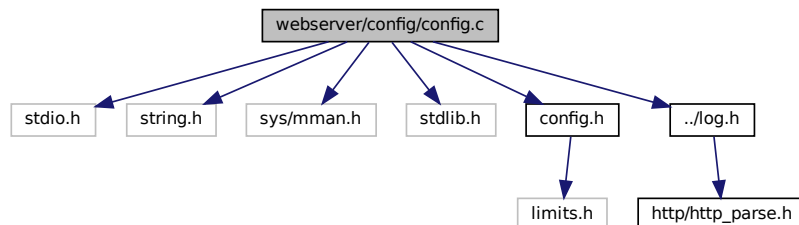
- [webserver/stats/stats.h](#)

Chapter 4

File Documentation

4.1 weserver/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

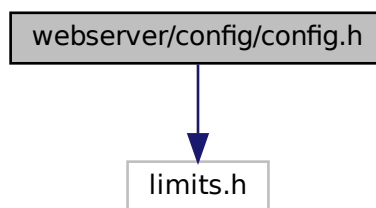
```
server_config* shared_mem_config
```

shared memory zone for access config in the whole application

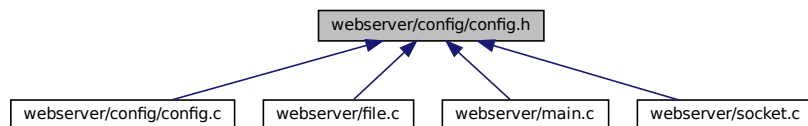
4.2 webserver/config/config.h File Reference

```
#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

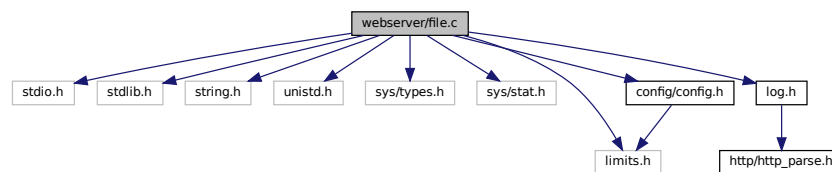
```
#include <stdio.h>  
#include <stdlib.h>
```

```

#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <limits.h>
#include <libgen.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](#) (void)

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 (
    void )
```

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
-------------	----------------------

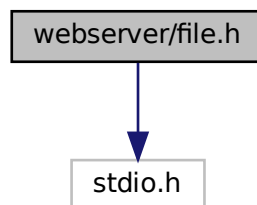
Returns

the mime type of the file

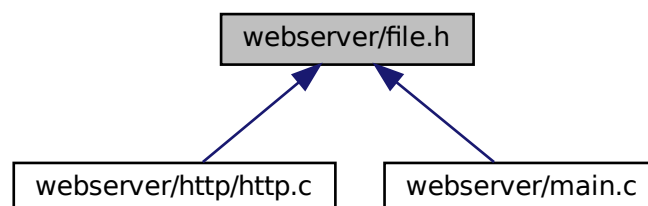
4.4 webserver/file.h File Reference

```
#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](#) (void)

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 (
    void )
```

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
-------------	----------------------

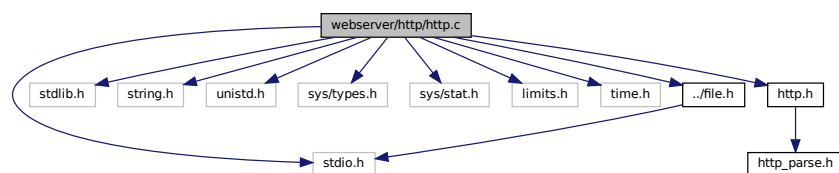
Returns

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 "../log.h"
```

Include dependency graph for http.c:



Functions

- void [skip_and_save_headers](#) (FILE *client, [http_request](#) *request)
- int [check_host_header](#) ([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 [check_host_header\(\)](#)

```
int check_host_header (
    http\_request * request )
```

check host header

Parameters

<i>request</i>	the request to check headers
----------------	------------------------------

Returns

0 if everything ok, 1 otherwise

4.5.1.2 [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

4.5.1.3 [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

4.5.1.4 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

4.5.1.5 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

4.5.1.6 skip_and_save_headers()

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

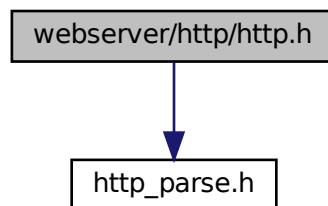
ignore headers of yhe requests

Parameters

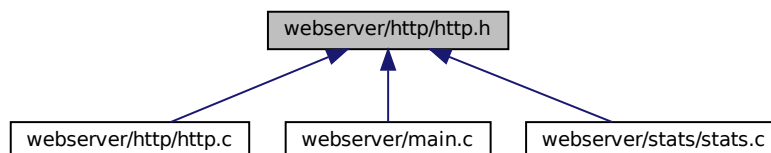
<i>client</i>	request stream
<i>request</i>	request we parse

4.6 webserver/http/http.h File Reference

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



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



Functions

- void `skip_and_save_headers` (FILE **client*, `http_request` **request*)

- int [check_host_header](#) ([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 [check_host_header\(\)](#)

```
int check_host_header (
    http\_request * request )
```

check host header

Parameters

<i>request</i>	the request to check headers
----------------	------------------------------

Returns

0 if everything ok, 1 otherwise

4.6.1.2 [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

4.6.1.3 [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

4.6.1.4 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

4.6.1.5 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

4.6.1.6 skip_and_save_headers()

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

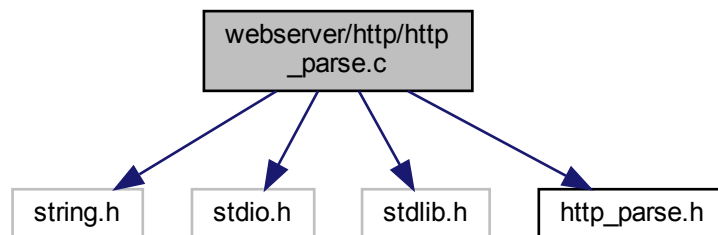
ignore headers of yhe requests

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 <stdlib.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

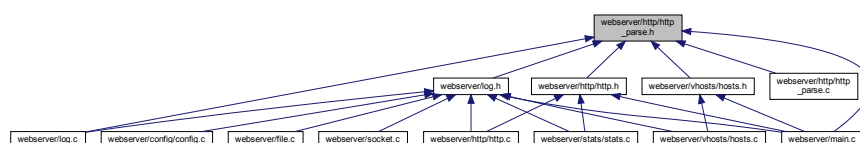
<i>request_line</i>	main request line
<i>request</i>	request to parse

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

- void [init_request](#) ([http_request](#) *request)
- 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 `init_request()`

```
void init_request (
    http\_request * request )
```

4.8.3.2 `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 http_request 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

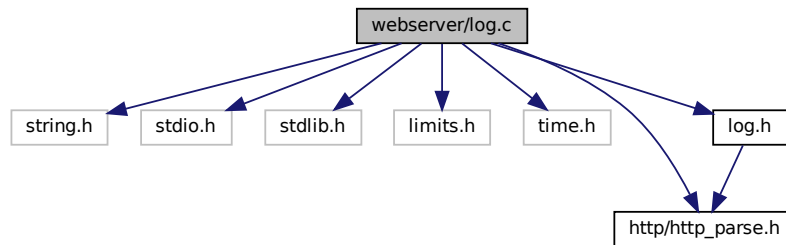
Returns

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

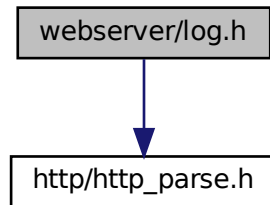
```
FILE* log_requests
```

the log file for requests

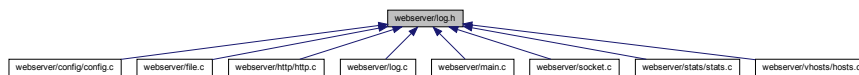
4.10 webserver/log.h File Reference

```
#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 "vhosts/hosts.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]
```

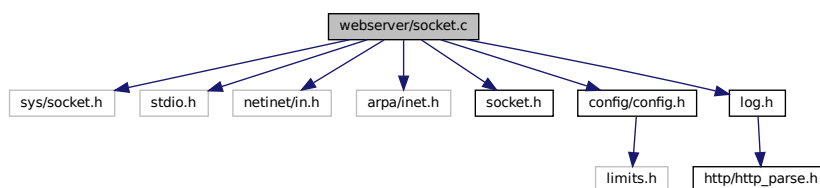
the root directory of the application

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

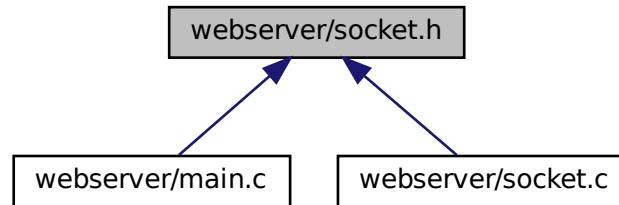
<i>port</i>	the port to listen
-------------	--------------------

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
-------------	--------------------

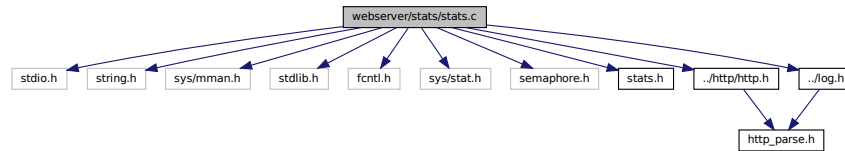
Returns

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

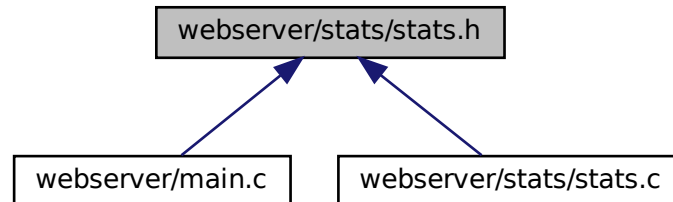
4.14.2.1 shared_memory

`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

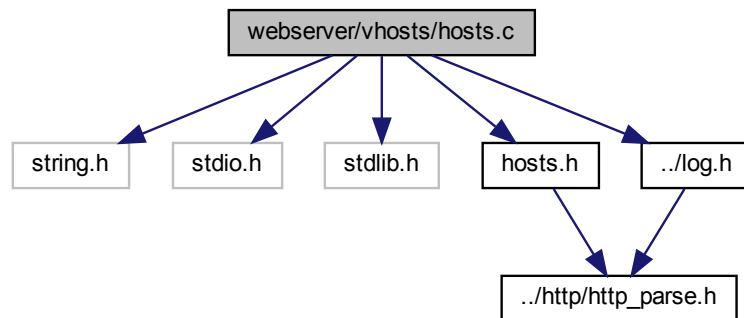
4.15.2.1 `shared_semaphore`

```
sem_t* shared_semaphore
```

semaphore to avoid concurrent access to the stats

4.16 webserv/vhosts/hosts.c File Reference

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



Functions

- char * [read_host_file](#) ([http_request](#) *request)
- char * [get_vhost_root](#) ([http_request](#) *request)

4.16.1 Function Documentation

4.16.1.1 `get_vhost_root()`

```
char* get_vhost_root (
    http_request * request )
```

get the root dir for a specific virtual host

Parameters

<i>request</i>	the request to parse host header
----------------	----------------------------------

Returns

the root for the virtual host

4.16.1.2 read_host_file()

```
char * read_host_file (
    http_request * request )
```

read virtual host config file and search for the host wanted in the request

Parameters

<i>request</i>	the request to check host header
----------------	----------------------------------

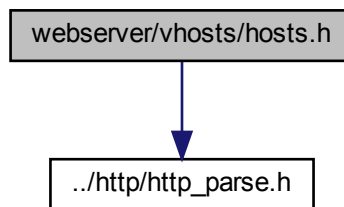
Returns

the root dir requested by the client

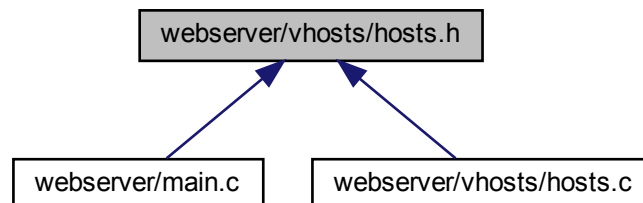
4.17 webserver/vhosts/hosts.h File Reference

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

Include dependency graph for hosts.h:



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



Functions

- char * [get_vhost_root](#) ([http_request](#) *request)

4.17.1 Function Documentation

4.17.1.1 [get_vhost_root\(\)](#)

```
char* get_vhost_root (
    http\_request * request )
```

get the root dir for a specific virtual host

Parameters

<i>request</i>	the request to parse host header
----------------	----------------------------------

Returns

the root for the virtual host

Index

- check_and_open
 - file.c, [13](#)
 - file.h, [17](#)
- check_host_header
 - http.c, [20](#)
 - http.h, [23](#)
- check_root
 - file.c, [13](#)
 - file.h, [17](#)
- child_handler
 - main.c, [35](#)
- client_ip
 - log.h, [34](#)
- 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, [29](#)
 - log.h, [32](#)
- create_requests_logs_file
 - log.c, [29](#)
 - log.h, [33](#)
- create_server
 - socket.c, [37](#)
 - socket.h, [38](#)
- 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, [23](#)
- get_file_size
 - file.c, [15](#)
 - file.h, [18](#)
- get_log_errors
 - log.c, [30](#)
 - log.h, [33](#)
- get_log_requests
 - log.c, [30](#)
 - log.h, [33](#)
- get_mime_type
 - file.c, [15](#)
 - file.h, [19](#)
- get_stats
 - stats.c, [39](#)
 - stats.h, [41](#)
- get_vhost_root
 - hosts.c, [42](#)
 - hosts.h, [44](#)
- headers
 - http_request, [5](#)
- hosts.c
 - get_vhost_root, [42](#)
 - read_host_file, [42](#)
- hosts.h
 - get_vhost_root, [44](#)
- http.c
 - check_host_header, [20](#)
 - get_date_http_format, [20](#)
 - rewrite_target, [20](#)
 - send_response, [21](#)
 - send_status, [21](#)
 - skip_and_save_headers, [21](#)

- http.h
 - check_host_header, [23](#)
 - get_date_http_format, [23](#)
 - rewrite_target, [23](#)
 - send_response, [24](#)
 - send_status, [24](#)
 - skip_and_save_headers, [24](#)
- HTTP_GET
 - http_parse.h, [27](#)
- HTTP_HEAD
 - http_parse.h, [27](#)
- http_major
 - http_request, [5](#)
- http_method
 - http_parse.h, [27](#)
- http_minor
 - http_request, [5](#)
- http_parse.c
 - in_range, [25](#)
 - min, [26](#)
 - parse_http_request, [26](#)
- http_parse.h
 - HTTP_GET, [27](#)
 - HTTP_HEAD, [27](#)
 - http_method, [27](#)
 - HTTP_UNSUPPORTED, [27](#)
 - init_request, [28](#)
 - MAX_TARGET_SIZE, [27](#)
 - parse_http_request, [28](#)
- 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_request
 - http_parse.h, [28](#)
- init_signals
 - main.c, [35](#)
- init_stats
 - stats.c, [39](#)
 - stats.h, [41](#)
- ko_400
 - web_stats, [7](#)
- ko_403
 - web_stats, [7](#)
- ko_404
 - web_stats, [8](#)
- ko_405
 - web_stats, [8](#)
- listen_addr
 - server_config, [6](#)
- log.c
 - create_errors_logs_file, [29](#)
 - create_requests_logs_file, [29](#)
 - get_log_errors, [30](#)
 - get_log_requests, [30](#)
 - log_errors, [31](#)
 - log_requests, [31](#)
 - write_error, [30](#)
 - write_request, [31](#)
- log.h
 - client_ip, [34](#)
 - create_errors_logs_file, [32](#)
 - create_requests_logs_file, [33](#)
 - get_log_errors, [33](#)
 - get_log_requests, [33](#)
 - write_error, [33](#)
 - write_request, [34](#)
- log_errors
 - log.c, [31](#)
- log_requests
 - log.c, [31](#)
- main
 - main.c, [35](#)
- main.c
 - child_handler, [35](#)
 - init_signals, [35](#)
 - main, [35](#)
 - respond_client, [36](#)
 - root, [36](#)
- MAX_TARGET_SIZE
 - http_parse.h, [27](#)
- method
 - http_request, [6](#)
- mimes_file
 - server_config, [6](#)
- min
 - http_parse.c, [26](#)
- ok_200
 - web_stats, [8](#)
- parse_http_request
 - http_parse.c, [26](#)
 - http_parse.h, [28](#)
- port
 - server_config, [7](#)
- read_host_file
 - hosts.c, [42](#)
- respond_client
 - main.c, [36](#)
- rewrite_target
 - http.c, [20](#)
 - http.h, [23](#)
- root
 - main.c, [36](#)

- send_response
 - http.c, 21
 - http.h, 24
- send_stats
 - stats.c, 39
 - stats.h, 41
- send_status
 - http.c, 21
 - http.h, 24
- served_connections
 - web_stats, 8
- served_requests
 - web_stats, 8
- server_config, 6
 - listen_addr, 6
 - mimes_file, 6
 - port, 7
- shared_mem_config
 - config.c, 10
- shared_memory
 - stats.c, 40
- shared_semaphore
 - stats.h, 41
- skip_and_save_headers
 - http.c, 21
 - http.h, 24
- socket.c
 - create_server, 37
- socket.h
 - create_server, 38
- stats.c
 - get_stats, 39
 - init_stats, 39
 - send_stats, 39
 - shared_memory, 40
- stats.h
 - get_stats, 41
 - init_stats, 41
 - send_stats, 41
 - shared_semaphore, 41
- target
 - http_request, 6
- web_stats, 7
 - ko_400, 7
 - ko_403, 7
 - 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, 25
- webserver/http/http_parse.h, 26
- webserver/log.c, 28
- webserver/log.h, 32
- webserver/main.c, 34
- webserver/socket.c, 36
- webserver/socket.h, 38
- webserver/stats/stats.c, 38
- webserver/stats/stats.h, 40
- webserver/vhosts/hosts.c, 42
- webserver/vhosts/hosts.h, 43
- write_error
 - log.c, 30
 - log.h, 33
- write_request
 - log.c, 31
 - log.h, 34