My Awesome Server

Generated by Doxygen 1.8.17

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 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]
- char log_dir [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 log_dir

```
char server_config::log_dir[PATH_MAX]
```

3.2.2.3 mimes_file

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

mime types file path

3.2.2.4 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

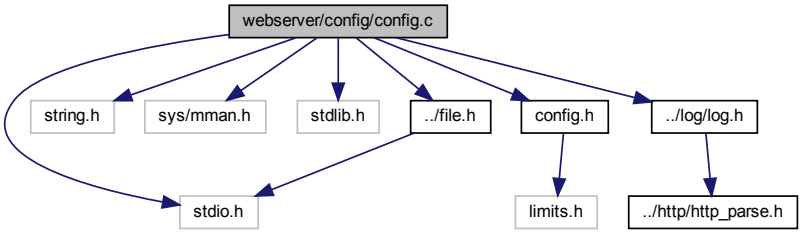
Chapter 4

File Documentation

4.1 webserver/config/config.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <sys/mman.h>
#include <stdlib.h>
#include "config.h"
#include "../log/log.h"
#include "../file.h"
```

Include dependency graph for config.c:



Functions

- int init_config ()
- int get_config_from_file (void)
- 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 (
            void  )
```

open the config file and read it

Parameters

abs_path	absolute path of the application

Returns

0 on success, 1 on error

4.1.1.3 init_config()

```
int init_config ( )
```

Init the configuration of the server

Parameters

abs_path	absolute path of the app

Returns

0 on success, 1 on error

4.1.2 Variable Documentation

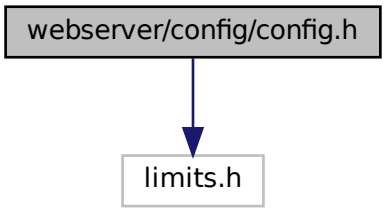
4.1.2.1 shared_mem_config

```
server_config* shared_mem_config
```

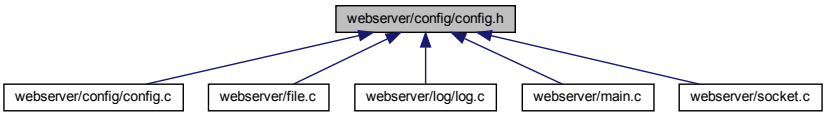
shared memory zone for access config in the whole application

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 ()
- int get_config_from_file ()
- 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 (
            void  )
```

open the config file and read it

Parameters

abs_path	absolute path of the application

Returns

0 on success, 1 on error

4.2.1.3 init_config()

```
int init_config ( )
```

Init the configuration of the server

Parameters

abs_path	absolute path of the app

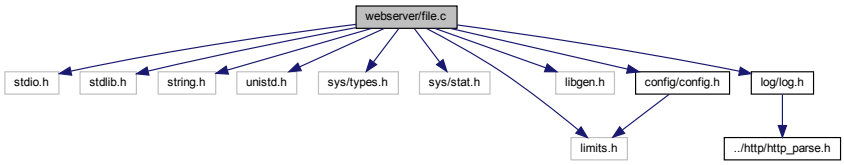
Returns

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/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

target	the target of the request
document_root	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

root	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

in	the file to read
out	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

buffer	buffer to store data
size	the size of data we read
stream	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

argv0	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

fd	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

name	the name of the file

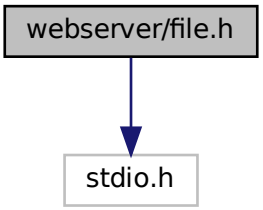
Returns

the mime type of the file

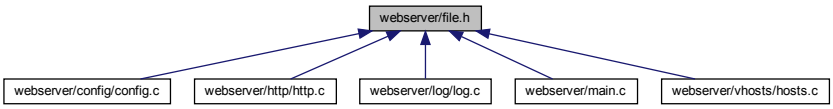
4.4 webserver/file.h File Reference

```
#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

target	the target of the request
document_root	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

root	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

in	the file to read
out	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

buffer	buffer to store data
size	the size of data we read
stream	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

argv0	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

fd	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

name	the name of the file

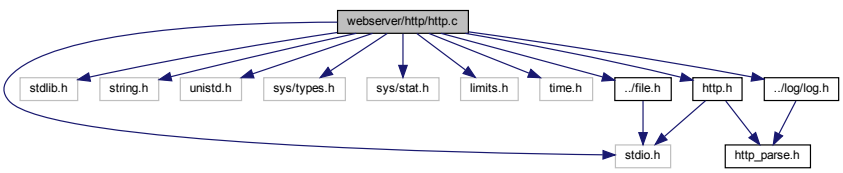
Returns

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/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

request	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

target	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

client	stream to write the response
code	the HTTP code of the response
reason_phrase	the reason phrase of the response
message_body	the body of the response
size	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

client	stream to send data
code	HTTP code of the response
reason_phrase	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

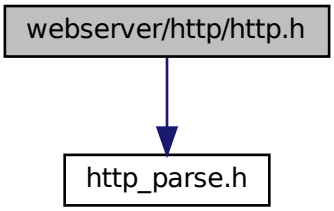
Parameters

client	request stream
request	request we parse

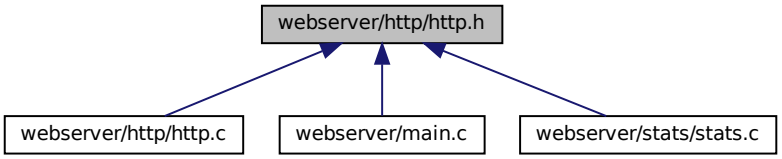
4.6 webserver/http/http.h File Reference

```
#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

request	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

target	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

client	stream to write the response
code	the HTTP code of the response
reason_phrase	the reason phrase of the response
message_body	the body of the response
size	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

client	stream to send data
code	HTTP code of the response
reason_phrase	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

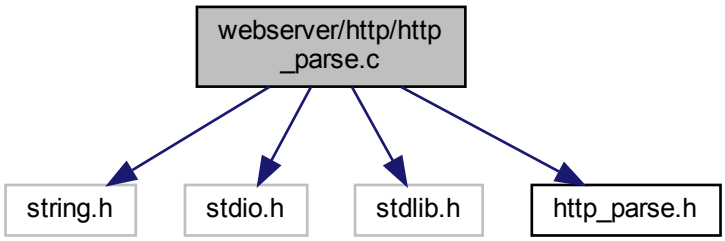
Parameters

client	request stream
request	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)
- char * get_method (enum http_method method)

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 get_method()

```
char* get_method (
             enum http_method method )
```

get the string name of the http method enum

Parameters

method	the method enum to check

Returns

the text name of the method

4.7.2.2 parse_http_request()

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

function to parse the request

Parameters

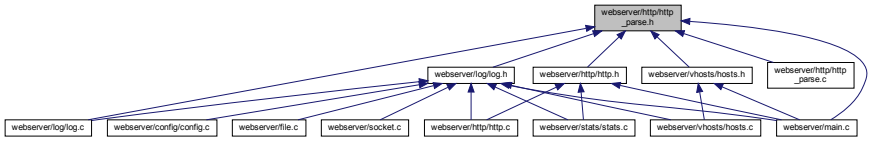
request_line	main request line
request	request to parse

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)
- char * get_method (enum http_method method)

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 get_method()

```
char* get_method (
            enum http_method method )
```

get the string name of the http method enum

Parameters

method	the method enum to check

Returns

the text name of the method

4.8.3.2 init_request()

```
void init_request (
            http_request * request )
```

4.8.3.3 parse_http_request()

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

Parses a http request line.

Parameters

in	request_line the line to parse (as a null terminated string)
out	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

request_line	main request line
request	request to parse

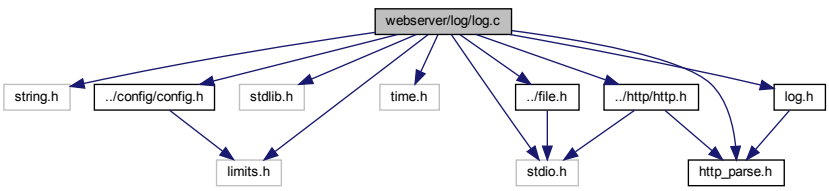
Returns

1 on success, 0 on error

4.9 webserver/log/log.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <time.h>
#include "../http/http.h"
#include "../http/http_parse.h"
#include "log.h"
#include "../config/config.h"
#include "../file.h"
```

Include dependency graph for log.c:



Functions

- void create_requests_logs_file (void)
- void create_errors_logs_file (void)
- 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 (
            void  )
```

init the errors log file

4.9.1.2 create_requests_logs_file()

```
void create_requests_logs_file (
            void  )
```

init the requests log file

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

log_file	the log file
request	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

log_file	the log file
request	the request to write
code	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

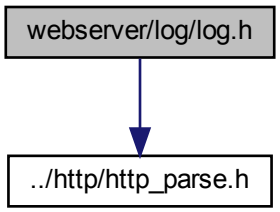
4.9.2.2 log_requests

```
FILE* log_requests
```

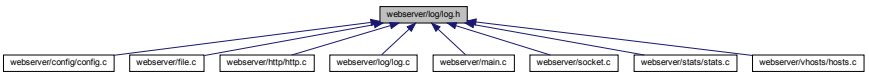
the log file for requests

4.10 webserver/log/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 (void)
- void create_errors_logs_file (void)
- 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 (
            void  )
```

init the errors log file

4.10.1.2 create_requests_logs_file()

```
void create_requests_logs_file (
            void  )
```

init the requests log file

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

log_file	the log file
request	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

log_file	the log file
request	the request to write
code	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/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

argc	arguments counter
argv	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

socket_client	the socket client

4.11.2 Variable Documentation

4.11.2.1 root

```
char root[PATH_MAX]
```

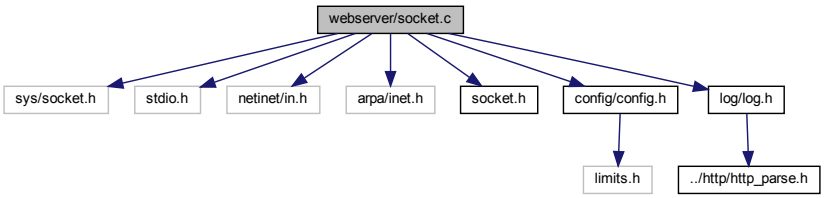
the root directory of the application

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/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

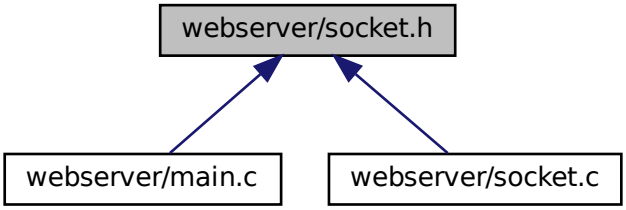
port	the port to listen

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

port	the port to listen

Returns

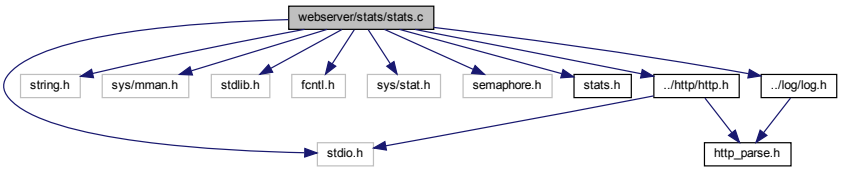
the server socket

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/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

client	the socket of the client

4.14.2 Variable Documentation

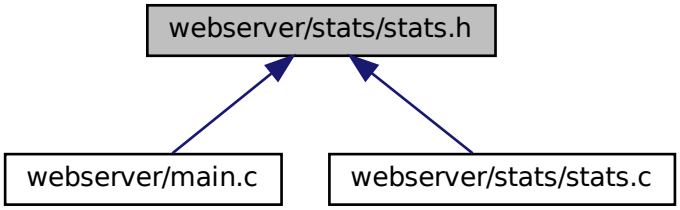
4.14.2.1 shared_memory

```
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

client	the socket of the client

4.15.2 Variable Documentation

4.15.2.1 shared_semaphore

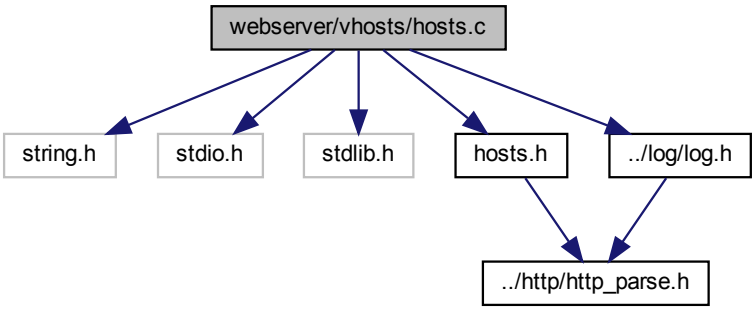
```
sem_t* shared_semaphore
```

semaphore to avoid concurrent access to the stats

4.16 webserver/vhosts/hosts.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "../file.h"
#include "hosts.h"
#include "../log/log.h"
```

Include dependency graph for hosts.c:



Functions

- 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 )
```

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

Parameters

request	the request to parse host header

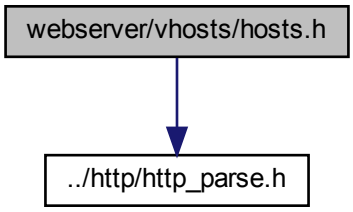
Returns

the root for the virtual host

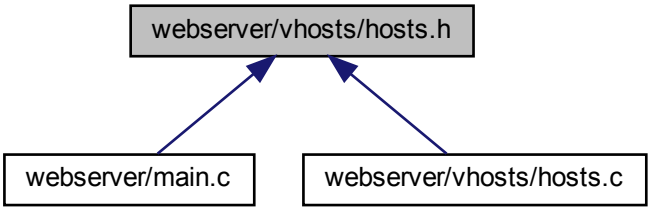
4.17 webserver/vhosts/hosts.h File Reference

```
#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 )
```

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

Parameters

request	the request to parse host header

Returns

the root for the virtual host

Index