Final Project - CE156 - Proxy Server”

Generated by Doxygen 1.8.1.2

Mon Jun 9 2014 17:55:02

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 http_parser.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdarg.h>
#include <dirent.h>
#include <sys/stat.h>
#include "http_parser.h"
#include "utils.h"
```

Macros

- #define SUCCESS 0
- #define FAILURE 1

Functions

- size_t count_newlines (char buffer[ ], size_t bufferlen)

  This will count the number of lines in an http message.
- void parse_http_header (char ∗lines[ ], char ∗buf, size_t numlines)

  Returns a an array with the http header broken into lines for further parsing.
- void parse_http_header_line (char ∗pieces[ ], char ∗line, size_t pieceslen)

  Parses one line from an http header.
- void free_parse_allocs (char ∗allocations[ ], int len)

  Frees memory allocations from parsing functions.

2.1.1 Macro Definition Documentation

2.1.1.1 #define FAILURE 1

2.1.1.2 #define SUCCESS 0

2.1.2 Function Documentation

2.1.2.1 size_t count_newlines (char *buffer[ ]*, size_t *bufferlen*)

This will count the number of lines in an http message.

Parameters

buffer	The buffer that is filled with the http message.
bufferlen	The length of the buffer passed in.

Returns

The numnber of newline characters.

2.1.2.2 void free_parse_allocs (char ∗ *allocations[ ]*, int *len*)

Frees memory allocations from parsing functions.

Parameters

allocations	An array filled with memory from the heap.
len	The length of the array.

2.1.2.3 void parse_http_header (char ∗ *lines[ ]*, char ∗ *buf*, size_t *numlines*)

Returns a an array with the http header broken into lines for further parsing.

Uses malloc so the array must free each element.

Parameters

buf	The http header.
numlines	The number of lines in the header.
lines	An array to fill with lines;

2.1.2.4 void parse_http_header_line (char ∗ *pieces[ ]*, char ∗ *line*, size_t *pieceslen*)

Parses one line from an http header.

Mallocs a slot in an array for each token. Must be freed after use.

Parameters

pieces	Array of char pointers that will get filled in with tokens.
line	The line to tokenize.
pieceslen	The length of the pieces array.

2.2 http_parser.h File Reference

This file contains the http header and body parsing functions.

Functions

- size_t count_newlines (char buffer[ ], size_t bufferlen)

This will count the number of lines in an http message.
- void parse_http_header (char ∗lines[ ], char ∗buf, size_t numlines)

  Returns a an array with the http header broken into lines for further parsing.
- void parse_http_header_line (char ∗pieces[ ], char ∗line, size_t pieceslen)

  Parses one line from an http header.
- void free_parse_allocs (char ∗allocations[ ], int len)

  Frees memory allocations from parsing functions.

2.2.1 Detailed Description

This file contains the http header and body parsing functions.

2.2.2 Function Documentation

2.2.2.1 size_t count_newlines (char *buffer[ ],* size_t *bufferlen*)

This will count the number of lines in an http message.

Parameters

buffer	The buffer that is filled with the http message.
bufferlen	The length of the buffer passed in.

Returns

The numnber of newline characters.

2.2.2.2 void free_parse_allocs (char ∗ *allocations[ ],* int *len*)

Frees memory allocations from parsing functions.

Parameters

allocations	An array filled with memory from the heap.
len	The length of the array.

2.2.2.3 void parse_http_header (char ∗ *lines[ ],* char ∗ *buf,* size_t *numlines*)

Returns a an array with the http header broken into lines for further parsing.

Uses malloc so the array must free each element.

Parameters

buf	The http header.
numlines	The number of lines in the header.
lines	An array to fill with lines;

2.2.2.4 void parse_http_header_line (char ∗ *pieces[ ],* char ∗ *line,* size_t *pieceslen*)

Parses one line from an http header.

Mallocs a slot in an array for each token. Must be freed after use.

Parameters

pieces	Array of char pointers that will get filled in with tokens.
line	The line to tokenize.
pieceslen	The length of the pieces array.

2.3 http_response.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdarg.h>
#include <dirent.h>
#include <sys/stat.h>
#include "http_response.h"
#include "utils.h"
```

Macros

- #define SUCCESS 0
- #define FAILURE 1

Functions

- void forbidden_response (char ∗buf, size_t buflen)
- void unimplmented_response (char ∗buf, size_t buflen)
- char ∗ http_response (const uint status)

  This function will return a formatted http response for the given status.

2.3.1 Macro Definition Documentation

2.3.1.1 #define FAILURE 1

2.3.1.2 #define SUCCESS 0

2.3.2 Function Documentation

2.3.2.1 void forbidden_response (char ∗ *buf,* size_t *buflen*)

2.3.2.2 char∗ http_response (const uint *status*)

This function will return a formatted http response for the given status.

Must be freed after use due to allocated on the heap.

Parameters

status	The http status to respond to a request with.

Returns

Returns a formatted http response or null if not implemented.

2.3.2.3 void unimplmented_response (char ∗ *buf,* size_t *buflen*)

2.4 http_response.h File Reference

This file contains the http header and body response functions.

Functions

- char ∗ http_response (const uint status)

  This function will return a formatted http response for the given status.

2.4.1 Detailed Description

This file contains the http header and body response functions.

2.4.2 Function Documentation

2.4.2.1 char∗ http_response (const uint *status*)

This function will return a formatted http response for the given status.

Must be freed after use due to allocated on the heap.

Parameters

status	The http status to respond to a request with.

Returns

Returns a formatted http response or null if not implemented.

2.5 log.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdarg.h>
#include <dirent.h>
#include <sys/stat.h>
#include <time.h>
#include "log.h"
#include "utils.h"
```

Macros

- #define SUCCESS 0

- #define FAILURE 1

Functions

- void get_current_time_formatted (char ∗formatted_time, int len)

    This functions fills the buffer passed in with formatted time string.
- void log_request (const char ∗time, const char ∗req, const char ∗version, const char ∗host, const char ∗uri, const char ∗ip, const char ∗action, const char ∗errors)

    Writes the following fields of the client request to the log file named, proxy.log.

2.5.1 Macro Definition Documentation

2.5.1.1 #define FAILURE 1

2.5.1.2 #define SUCCESS 0

2.5.2 Function Documentation

2.5.2.1 void get_current_time_formatted (char ∗ *buffer,* int *len*)

This functions fills the buffer passed in with formatted time string.

Parameters

buffer	A char buffer of size 64 bytes or greater.
len	The length of the buffer

2.5.2.2 void log_request (const char ∗ *time,* const char ∗ *req,* const char ∗ *version,* const char ∗ *host,* const char ∗ *uri,* const char ∗ *ip,* const char ∗ *action,* const char ∗ *errors*)

Writes the following fields of the client request to the log file named, proxy.log.

Parameters

time	The date and time of the request.
req	The type of request.
version	The version of http.
host	The requesting hostname.
uri	The uri of the request.
ip	The server ip address requested.
action	The action that the proxy server took.
errors	Any errors that occured.

2.6 log.h File Reference

This file contains all server logging api calls.

Functions

- void get_current_time_formatted (char ∗buffer, int len)

    This functions fills the buffer passed in with formatted time string.

- void log_request (const char ∗time, const char ∗req, const char ∗version, const char ∗host, const char ∗uri, const char ∗ip, const char ∗action, const char ∗errors)

  Writes the following fields of the client request to the log file named, proxy.log.

2.6.1 Detailed Description

This file contains all server logging api calls.

2.6.2 Function Documentation

2.6.2.1 void get_current_time_formatted (char ∗ *buffer,* int *len*)

This functions fills the buffer passed in with formatted time string.

Parameters

buffer	A char buffer of size 64 bytes or greater.
len	The length of the buffer

2.6.2.2 void log_request (const char ∗ *time,* const char ∗ *req,* const char ∗ *version,* const char ∗ *host,* const char ∗ *uri,* const char ∗ *ip,* const char ∗ *action,* const char ∗ *errors*)

Writes the following fields of the client request to the log file named, proxy.log.

Parameters

time	The date and time of the request.
req	The type of request.
version	The version of http.
host	The requesting hostname.
uri	The uri of the request.
ip	The server ip address requested.
action	The action that the proxy server took.
errors	Any errors that occured.

2.7 proxy_server.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <errno.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <sys/select.h>
#include <time.h>
#include <sys/ioctl.h>
#include <pthread.h>
#include "utils.h"
#include "http_parser.h"
#include "http_response.h"
#include "log.h"
#include "siteblock.h"
```

Macros

- #define SUCCESS 0
- #define FAILURE 1
- #define TRUE 1
- #define FALSE 0
- #define BUFLEN 8∗1024

Typedefs

- typedef struct sockaddr sockaddr
- typedef struct sockaddr_in sockaddr_in

Functions

- void ∗ handle_client_request (void ∗clisock)
- void close_client (int clisock, fd_set ∗master)
- int main (int argc, char ∗∗argv)

2.7.1 Macro Definition Documentation

2.7.1.1 #define BUFLEN 8∗1024

2.7.1.2 #define FAILURE 1

2.7.1.3 #define FALSE 0

2.7.1.4 #define SUCCESS 0

2.7.1.5 #define TRUE 1

2.7.2 Typedef Documentation

2.7.2.1 typedef struct sockaddr sockaddr

2.7.2.2 typedef struct sockaddr_in sockaddr_in

2.7.3 Function Documentation

2.7.3.1 void close_client (int *clisock*, fd_set ∗ *master*)

2.7.3.2 void ∗ handle_client_request (void ∗ *clisock*)

2.7.3.3 int main (int *argc*, char ∗∗ *argv*)

2.8 siteblock.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdarg.h>
#include <dirent.h>
#include <sys/stat.h>
#include <time.h>
#include "siteblock.h"
#include "utils.h"
```

Macros

- #define SUCCESS 0
- #define FAILURE 1

Functions

- int allowed_site (const char ∗site)

  Tells the user if the site is allowed to be visted.

2.8.1 Macro Definition Documentation

2.8.1.1 #define FAILURE 1

2.8.1.2 #define SUCCESS 0

2.8.2 Function Documentation

2.8.2.1 int allowed_site (const char ∗ *site*)

Tells the user if the site is allowed to be visted.

If no file named parental_controls.log, or no lines in that file all sites are allowed.

Parameters

site	The site the user wishes to check permissons on.

Returns

0 if allowed or -1 if not allowed to be visited.

2.9 siteblock.h File Reference

This file contains all server site blocking api calls.

Functions

- int allowed_site (const char ∗site)

  Tells the user if the site is allowed to be visted.

2.9.1 Detailed Description

This file contains all server site blocking api calls.

2.9.2 Function Documentation

2.9.2.1 int allowed_site (const char ∗ *site*)

Tells the user if the site is allowed to be visted.

If no file named parental_controls.log, or no lines in that file all sites are allowed.

Parameters

site	The site the user wishes to check permissons on.

Returns

0 if allowed or -1 if not allowed to be visited.

2.10 utils.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdarg.h>
#include <dirent.h>
#include <sys/stat.h>
#include "utils.h"
```

Macros

- #define SUCCESS 0

- #define FAILURE 1

Functions

- void check_socket (int fd)

  Checks the return value of the socket(3) networking api call for any errors and prints messages and sets the exit status accordingly.
- void check_connection (int x)

  Checks the return value of the connect(3) networking api call for any errors and prints messages and sets the exit status accordingly.
- FILE ∗ retrieve_file (const char ∗restrict filename, const char ∗restrict mode)

  Checks the current directory for the file filename.
- int get_file_size (FILE ∗restrict file)

  Takes a file and gives back the size of the file.
- unsigned char ∗ serialize_int (unsigned char ∗buffer, unsigned int val)

  Serializes an int into a unsigned char.
- unsigned char ∗ serialize_data (unsigned char ∗buffer, char buf[], int len)

  Serializes an char array into a unsigned char array.
- unsigned char ∗ deserialize_int (unsigned char ∗buffer, unsigned int ∗val)

  Deserializes an int into a unsigned char.
- unsigned char ∗ deserialize_data (unsigned char ∗buffer, char buf[], int len)

  Deserializes an char array into a unsigned char array.
- void null_array (char ∗array[], int len)

  Points each element to null in the array.
- void debugprintf (char ∗format,...)

2.10.1 Macro Definition Documentation

2.10.1.1 #define FAILURE 1

2.10.1.2 #define SUCCESS 0

2.10.2 Function Documentation

2.10.2.1 void check_connection (int *val*)

Checks the return value of the connect(3) networking api call for any errors and prints messages and sets the exit status accordingly.

Parameters

val	The return value from the connect(3) call.

2.10.2.2 void check_socket (int *fd*)

Checks the return value of the socket(3) networking api call for any errors and prints messages and sets the exit status accordingly.

Parameters

fd	The file descriptor returned by the socket(3) call.

2.10.2.3 void debugprintf (char ∗ *format,* ...)

2.10.2.4 unsigned char∗ deserialize_data (unsigned char ∗ *buffer,* char *buf[ ],* int *len*)

Deserializes an char array into a unsigned char array.

Parameters

buffer	The array to get the data from.
buf	The buffer to save it.
len	Then length of buf.

Returns

A pointer to the next free space in the buffer.

2.10.2.5 unsigned char∗ deserialize_int (unsigned char ∗ *buffer,* unsigned int ∗ *val*)

Deserializes an int into a unsigned char.

Parameters

buffer	The array to get the data out of.
val	The value to save the data.

Returns

A pointer to the next free space in the buffer.

2.10.2.6 int get_file_size (FILE ∗restrict *filename*)

Takes a file and gives back the size of the file.

Parameters

filename	The file in which you want the size of.

Returns

The size of the file filename, or -1 if an error occurs. Errno will be set to the proper error.

2.10.2.7 void null_array (char ∗ *array[ ],* int *len*)

Points each element to null in the array.

Parameters

array	The array to null out.
len	The length of the array.

2.10.2.8 FILE∗ retrieve_file (const char ∗restrict *filename,* const char ∗restrict *mode*)

Checks the current directory for the file filename.

If file name is found retrieve_file will attepmt to open the file using fopen. If not an error message will be returned. fclose(3) must be called or memory leak will occur.

Parameters

filename	The file to be searched for and opened.
mode	The mode in which the file will be opened.

Returns

The file descriptor for the file if fopen succeeds. Otherwise NULL is returned if filename is not found, or if fopen fails.

2.10.2.9 unsigned char∗ serialize_data (unsigned char ∗ *buffer,* char *buf[ ],* int *len*)

Serializes an char array into a unsigned char array.

Parameters

buffer	The array to insert the data.
buf	The value to serialize.
len	Then length of buf.

Returns

A pointer to the next free space in the buffer.

2.10.2.10 unsigned char∗ serialize_int (unsigned char ∗ *buffer,* unsigned int *val*)

Serializes an int into a unsigned char.

Parameters

buffer	The array to insert the data.
val	The value to serialize.

Returns

A pointer to the next free space in the buffer.

2.11 utils.h File Reference

Macros

- #define DEBUGF(...) debugprintf (__VA_ARGS__)

  Allows for debugging print statements to be made and easily turned off for release build.

Enumerations

- enum bool { FALSE = 0, TRUE = 1 }

  A boolean data type created by an enum.

Functions

- void check_socket (int fd)

  Checks the return value of the socket(3) networking api call for any errors and prints messages and sets the exit status accordingly.
- void check_connection (int val)

  Checks the return value of the connect(3) networking api call for any errors and prints messages and sets the exit status accordingly.
- FILE ∗ retrieve_file (const char ∗restrict filename, const char ∗restrict mode)

  Checks the current directory for the file filename.
- int get_file_size (FILE ∗restrict filename)

  Takes a file and gives back the size of the file.
- unsigned char ∗ serialize_int (unsigned char ∗buffer, unsigned int val)

  Serializes an int into a unsigned char.
- unsigned char ∗ serialize_data (unsigned char ∗buffer, char buf[ ], int len)

  Serializes an char array into a unsigned char array.
- unsigned char ∗ deserialize_int (unsigned char ∗buffer, unsigned int ∗val)

  Deserializes an int into a unsigned char.
- unsigned char ∗ deserialize_data (unsigned char ∗buffer, char buf[ ], int len)

  Deserializes an char array into a unsigned char array.
- void null_array (char ∗array[ ], int len)

  Points each element to null in the array.
- void debugprintf (char ∗format,...)

2.11.1 Macro Definition Documentation

2.11.1.1 #define DEBUGF(...) debugprintf (__VA_ARGS__)

Allows for debugging print statements to be made and easily turned off for release build.

Parameters

format	The format string to format the print statement.

2.11.2 Enumeration Type Documentation

2.11.2.1 enum bool

A boolean data type created by an enum.

Enumerator:

FALSE

TRUE

2.11.3 Function Documentation

2.11.3.1 void check_connection (int *val*)

Checks the return value of the connect(3) networking api call for any errors and prints messages and sets the exit status accordingly.

Parameters

val	The return value from the connect(3) call.

2.11.3.2 void check_socket (int *fd*)

Checks the return value of the socket(3) networking api call for any errors and prints messages and sets the exit status accordingly.

Parameters

fd	The file descriptor returned by the socket(3) call.

2.11.3.3 void debugprintf (char ∗ *format*, ...)

2.11.3.4 unsigned char∗ deserialize_data (unsigned char ∗ *buffer*, char *buf[ ]*, int *len*)

Deserializes an char array into a unsigned char array.

Parameters

buffer	The array to get the data from.
buf	The buffer to save it.
len	Then length of buf.

Returns

A pointer to the next free space in the buffer.

2.11.3.5 unsigned char∗ deserialize_int (unsigned char ∗ *buffer*, unsigned int ∗ *val*)

Deserializes an int into a unsigned char.

Parameters

buffer	The array to get the data out of.
val	The value to save the data.

Returns

A pointer to the next free space in the buffer.

2.11.3.6 int get_file_size (FILE ∗restrict *filename*)

Takes a file and gives back the size of the file.

Parameters

filename	The file in which you want the size of.

Returns

The size of the file filename, or -1 if an error occurs. Errno will be set to the proper error.

2.11.3.7 void null_array (char ∗ *array[ ],* int *len*)

Points each element to null in the array.

Parameters

array	The array to null out.
len	The length of the array.

2.11.3.8 FILE∗ retrieve_file (const char ∗restrict *filename,* const char ∗restrict *mode*)

Checks the current directory for the file filename.

If file name is found retrieve_file will attepmt to open the file using fopen. If not an error message will be returned. fclose(3) must be called or memory leak will occur.

Parameters

filename	The file to be searched for and opened.
mode	The mode in which the file will be opened.

Returns

The file descriptor for the file if fopen succeeds. Otherwise NULL is returned if filename is not found, or if fopen fails.

2.11.3.9 unsigned char∗ serialize_data (unsigned char ∗ *buffer,* char *buf[ ],* int *len*)

Serializes an char array into a unsigned char array.

Parameters

buffer	The array to insert the data.
buf	The value to serialize.
len	Then length of buf.

Returns

A pointer to the next free space in the buffer.

2.11.3.10 unsigned char∗ serialize_int (unsigned char ∗ *buffer,* unsigned int *val*)

Serializes an int into a unsigned char.

Parameters

buffer	The array to insert the data.
val	The value to serialize.

Returns

A pointer to the next free space in the buffer.