EHN Group 12 Practical 1

Generated by Doxygen 1.8.13

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Module Documentation

4.1 Client

Macros

- #define MAX_REQ_LEN 255

  The maximum length of a request (characters).
- #define DEBUG 0

  Enable (1) or disable (0) client debugging.

Functions

- int main (int argc, char ∗argv[ ])
- void clear_buffer (char ∗buffer, int length)

4.1.1 Detailed Description

This module contains the full client program, which allows the user to connect to the specified server, request and download files, and display web pages in the terminal

4.1.2 Macro Definition Documentation

4.1.2.1 DEBUG

```
#define DEBUG 0
```

Enable (1) or disable (0) client debugging.

4.1.2.2 MAX_REQ_LEN

```
#define MAX_REQ_LEN 255
```

The maximum length of a request (characters).

4.1.3 Function Documentation

4.1.3.1 clear_buffer()

```
void clear_buffer (
             char * buffer,
             int length )
```

Clears a buffer up to a specified length.

Parameters

buffer	The buffer to be cleared.
length	The length up to which the buffer must be cleared.

4.1.3.2 main()

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

Sets up the client SSL connection, connects to the server and then displays or downloads requested files from the server

Parameters

argc	The number of arguments passes to the function.
argv	The values of the passes arguments as c-strings.

Returns

Successful or failed execution.

4.2 Server

Data Structures

- struct server_args

  This struct is passed as an argument to the server thread to allow multiple arguments to be passed.
- struct client_args

  This struct is passed as an argument to newly created client threads to allow multiple arguments to be passed.

Macros

- #define DEBUG 0

  Enable (1) or disable (0) server debugging.

Functions

- int main (int argc, char ∗argv[ ])
- void ∗ server_thread (void ∗ptr)
- void ∗ new_client_connection (void ∗ptr)
- pthread_t ∗ double_size (pthread_t ∗old_clients, int current_size)
- int write_page (BIO ∗bio, const char ∗page, const char ∗filename)
- int read_media ()
- char ∗ itoa (char ∗result, int number)

Variables

- char MediaItems [100][256]

  An array of c-strings to store the names of the files that can be downloaded.
- int numMediaItems = 0

  The number of files that can be downloaded.
- int SERVER_RUN

  Controls the execution of the server thread.

4.2.1 Detailed Description

This module contains the entire server program. The server starts an SSL server with the specified key and certificate files and listens for client connections. It also indexes all files stored in the Media_files folder and presents the list of items to the client to allow the client to download the files.

4.2.2 Macro Definition Documentation

4.2.2.1 DEBUG

```
#define DEBUG 0
```

Enable (1) or disable (0) server debugging.

4.2.3 Function Documentation

4.2.3.1 double_size()

```
pthread_t* double_size (
             pthread_t * old_clients,
             int current_size )
```

When the current clients array is full, create a new one with double the size.

Parameters

old_clients	The previous array of clients.
current_size	The previous size of the clients array.

Returns

A pointer to the new clients array.

4.2.3.2 itoa()

```
char* itoa (
             char * result,
             int number )
```

Convert between an integer and a c-string.

Parameters

result	The c-string to be used for the output.
number	The number to be converted

Returns

The same c-string used for the output.

4.2.3.3 main()

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

The main function sets up all the SSL functions, Certificates and starts the server.

Parameters

argc	The number of arguments passed to the function.
argv	The values of the passed arguments as c-strings.

Returns

Successful or failed execution.

Sets up the client SSL connection, connects to the server and then displays or downloads requested files from the server

Parameters

argc	The number of arguments passes to the function.
argv	The values of the passes arguments as c-strings.

Returns

Successful or failed execution.

4.2.3.4 new_client_connection()

```
void* new_client_connection (
             void * ptr )
```

This function is created as a new thread for every client that makes a request to the server.

Parameters

ptr	The client_args struct is passed as a void pointer.

Returns

Successful or failed execution.

4.2.3.5 read_media()

```
int read_media ( )
```

Read all the contents of the Media_files folder for use later in GET requests.

Returns

Successful or failed execution.

4.2.3.6 server_thread()

```
void* server_thread (
            void * ptr )
```

This function is created as a new thread and handles all client requests.

Parameters

ptr	The server_args struct is passed as a void pointer.

Returns

Successful or failed execution.

4.2.3.7 write_page()

```
int write_page (
            BIO * bio,
            const char * page,
            const char * filename )
```

Write an arbitrary file to the client.

Parameters

bio	A pointer to the client's SSL object.
page	The file to be written.
filename	The name of the file to be written.

Returns

Successful or failed execution.

4.2.4 Variable Documentation

4.2.4.1 MediaItems

```
char MediaItems[100][256]
```

An array of c-strings to store the names of the files that can be downloaded.

4.2.4.2 numMediaItems

```
int numMediaItems = 0
```

The number of files that can be downloaded.

4.2.4.3 SERVER_RUN

```
int SERVER_RUN
```

Controls the execution of the server thread.

Chapter 5

Data Structure Documentation

5.1 client_args Struct Reference

This struct is passed as an argument to newly created client threads to allow multiple arguments to be passed.

```
#include <Server.h>
```

Data Fields

- BIO ∗ abio

  The SSL object pointer.
- int thread_number

  The current thread number.

5.1.1 Detailed Description

This struct is passed as an argument to newly created client threads to allow multiple arguments to be passed.

5.1.2 Field Documentation

5.1.2.1 abio

```
BIO∗ client_args::abio
```

The SSL object pointer.

5.1.2.2 thread_number

```
int client_args::thread_number
```

The current thread number.

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

- Server.h

5.2 server_args Struct Reference

This struct is passed as an argument to the server thread to allow multiple arguments to be passed.

```
#include <Server.h>
```

Data Fields

- BIO ∗ acpt

    The SSL reception buffer.
- BIO ∗ abio

    The SSL object pointer.

5.2.1 Detailed Description

This struct is passed as an argument to the server thread to allow multiple arguments to be passed.

5.2.2 Field Documentation

5.2.2.1 abio

```
BIO* server_args::abio
```

The SSL object pointer.

5.2.2.2 acpt

```
BIO* server_args::acpt
```

The SSL reception buffer.

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

- Server.h

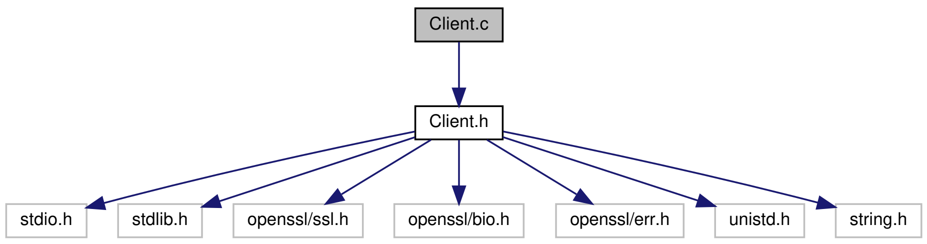
Chapter 6

File Documentation

6.1 Client.c File Reference

```
#include "Client.h"
```

Include dependency graph for Client.c:



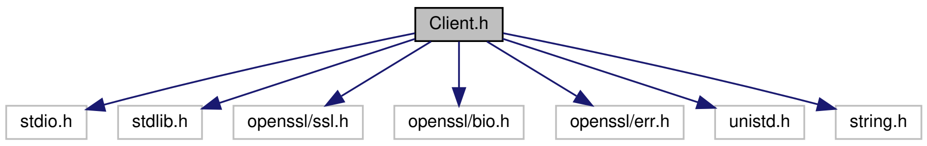
Functions

- int main (int argc, char ∗argv[ ])
- void clear_buffer (char ∗buffer, int length)

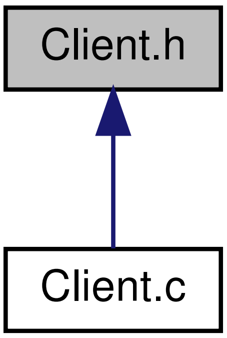
6.2 Client.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <openssl/ssl.h>
#include <openssl/bio.h>
#include <openssl/err.h>
#include <unistd.h>
#include <string.h>
```

Include dependency graph for Client.h:



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



Macros

- #define MAX_REQ_LEN 255

  The maximum length of a request (characters).
- #define DEBUG 0

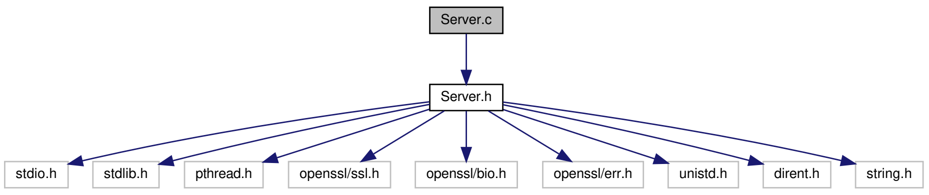
  Enable (1) or disable (0) client debugging.

Functions

- int main (int argc, char ∗argv[ ])
- void clear_buffer (char ∗buffer, int length)

6.3 Server.c File Reference

```
#include "Server.h"
```
Include dependency graph for Server.c:



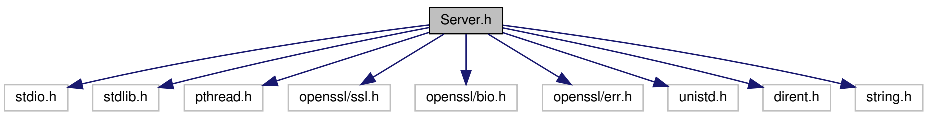
Functions

- int main (int argc, char ∗argv[ ])
- void ∗ server_thread (void ∗ptr)
- void ∗ new_client_connection (void ∗ptr)
- int read_media ()
- int write_page (BIO ∗bio, const char ∗page, const char ∗filename)
- pthread_t ∗ double_size (pthread_t ∗old_clients, int current_size)
- char ∗ itoa (char ∗result, int number)

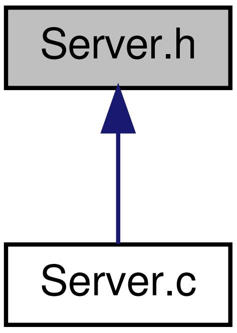
6.4 Server.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <openssl/ssl.h>
#include <openssl/bio.h>
#include <openssl/err.h>
#include <unistd.h>
#include <dirent.h>
#include <string.h>
```

Include dependency graph for Server.h:



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



Data Structures

- struct server_args

  This struct is passed as an argument to the server thread to allow multiple arguments to be passed.
- struct client_args

  This struct is passed as an argument to newly created client threads to allow multiple arguments to be passed.

Macros

- #define DEBUG 0

  Enable (1) or disable (0) server debugging.

Functions

- int main (int argc, char ∗argv[ ])
- void ∗ server_thread (void ∗ptr)
- void ∗ new_client_connection (void ∗ptr)
- pthread_t ∗ double_size (pthread_t ∗old_clients, int current_size)
- int write_page (BIO ∗bio, const char ∗page, const char ∗filename)
- int read_media ()
- char ∗ itoa (char ∗result, int number)

Variables

- char MediaItems [100][256]

  An array of c-strings to store the names of the files that can be downloaded.
- int numMediaItems = 0

  The number of files that can be downloaded.
- int SERVER_RUN

  Controls the execution of the server thread.