

echoserver

Generated by Doxygen 1.8.1.2

Fri Aug 30 2013 19:10:52

Contents

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	ServerTag Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	5
3.1.2.1	c_socket	5
4	File Documentation	7
4.1	echo_server.c File Reference	7
4.1.1	Detailed Description	8
4.1.2	Macro Definition Documentation	8
4.1.2.1	MAX_BUFFER_LEN	8
4.1.3	Function Documentation	8
4.1.3.1	echo_server	8
4.1.4	Variable Documentation	8
4.1.4.1	time_out_msg	8
4.1.4.2	time_out_secs	8
4.1.4.3	time_out_usecs	8
4.2	echo_server.h File Reference	9
4.2.1	Detailed Description	9
4.2.2	Function Documentation	9
4.2.2.1	echo_server	9
4.3	helper.c File Reference	10
4.3.1	Detailed Description	10
4.3.2	Macro Definition Documentation	10
4.3.2.1	MAX_BUFFER_SIZE	10
4.3.3	Function Documentation	10

4.3.3.1	print_errno_message	11
4.4	helper.h File Reference	11
4.4.1	Detailed Description	11
4.4.2	Macro Definition Documentation	12
4.4.2.1	DFPRINTF	12
4.4.2.2	DPRINTF	12
4.4.3	Function Documentation	12
4.4.3.1	print_errno_message	12
4.5	main.c File Reference	12
4.5.1	Detailed Description	13
4.5.2	Function Documentation	13
4.5.2.1	get_port_from_commandline	13
4.5.2.2	main	13
4.6	server.c File Reference	13
4.6.1	Detailed Description	14
4.6.2	Macro Definition Documentation	14
4.6.2.1	IPV6	14
4.6.3	Function Documentation	14
4.6.3.1	create_server_socket	14
4.6.3.2	start_server	14
4.6.4	Variable Documentation	15
4.6.4.1	backlog	15
4.7	server.h File Reference	15
4.7.1	Detailed Description	16
4.7.2	Macro Definition Documentation	16
4.7.2.1	DDECREMENT_THREAD_COUNT	16
4.7.2.2	DINCREMENT_THREAD_COUNT	16
4.7.3	Typedef Documentation	16
4.7.3.1	ServerTag	16
4.7.4	Function Documentation	17
4.7.4.1	create_server_socket	17
4.7.4.2	start_server	17
4.8	socket_helpers.c File Reference	17
4.8.1	Detailed Description	18
4.8.2	Function Documentation	18
4.8.2.1	socket_readline	18
4.8.2.2	socket_readline_timeout	18
4.8.2.3	socket_writeline	19
4.9	socket_helpers.h File Reference	19
4.9.1	Detailed Description	20

4.9.2	Function Documentation	20
4.9.2.1	socket_readline	20
4.9.2.2	socket_readline_timeout	20
4.9.2.3	socket_writeline	21

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

ServerTag	
Struct for passing to server threads	5

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

echo_server.c	Implementation of echo server functions	7
echo_server.h	Interface to echo server functions	9
helper.c	Implementation of helper functions	10
helper.h	Interface to helper functions	11
main.c	Main function for echoserver	12
server.c	Implementation of listening server functions	13
server.h	Interface to listening server functions	15
socket_helpers.c	Implementation of socket helper functions	17
socket_helpers.h	Interface to socket helper functions	19

Chapter 3

Data Structure Documentation

3.1 ServerTag Struct Reference

Struct for passing to server threads.

```
#include <server.h>
```

Data Fields

- int [c_socket](#)

3.1.1 Detailed Description

Struct for passing to server threads.

Contains a file descriptor for the connected socket, as the server obviously needs to know this.

3.1.2 Field Documentation

3.1.2.1 int ServerTag::c_socket

File descriptor for the connected socket

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

- [server.h](#)

Chapter 4

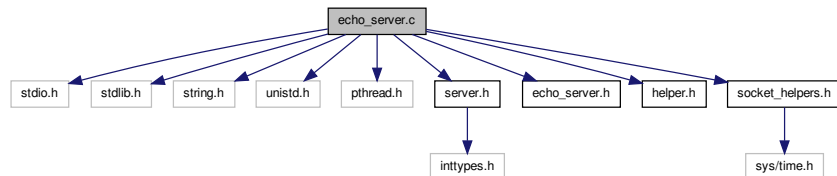
File Documentation

4.1 echo_server.c File Reference

Implementation of echo server functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include "server.h"
#include "echo_server.h"
#include "helper.h"
#include "socket_helpers.h"
```

Include dependency graph for echo_server.c:



Macros

- #define `MAX_BUFFER_LEN` 1024

Functions

- void * `echo_server` (void *arg)
Main echo server handler thread function.

Variables

- static const long `time_out_secs` = 5
File scope variable for default time out seconds.

- static const long `time_out_usecs` = 0
File scope variable for default time out microseconds.
- static const char `time_out_msg` [] = "Timeout - closing connection.\n"
File scope variable for timeout message.

4.1.1 Detailed Description

Implementation of echo server functions.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.1.2 Macro Definition Documentation

4.1.2.1 #define MAX_BUFFER_LEN 1024

4.1.3 Function Documentation

4.1.3.1 void* echo_server (void * arg)

Main echo server handler thread function.

Provides echo server service to a provided connected socket. The server loops and echoes any whole lines provided. The server will time-out after a pre-defined period, if no input, or if no more input, is received.

Parameters

<code>arg</code>	Pointer to a ServerTag struct
------------------	---

Returns

NULL

4.1.4 Variable Documentation

4.1.4.1 const char time_out_msg[] = "Timeout - closing connection.\n" [static]

File scope variable for timeout message.

4.1.4.2 const long time_out_secs = 5 [static]

File scope variable for default time out seconds.

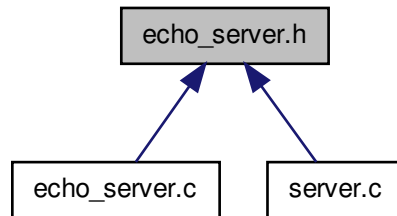
4.1.4.3 const long time_out_usecs = 0 [static]

File scope variable for default time out microseconds.

4.2 echo_server.h File Reference

Interface to echo server functions.

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



Functions

- void * [echo_server](#) (void *arg)
Main echo server handler thread function.

4.2.1 Detailed Description

Interface to echo server functions.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.2.2 Function Documentation

4.2.2.1 void* echo_server (void * arg)

Main echo server handler thread function.

Provides echo server service to a provided connected socket. The server loops and echoes any whole lines provided. The server will time-out after a pre-defined period, if no input, or if no more input, is received.

Parameters

<i>arg</i>	Pointer to a ServerTag struct
------------	---

Returns

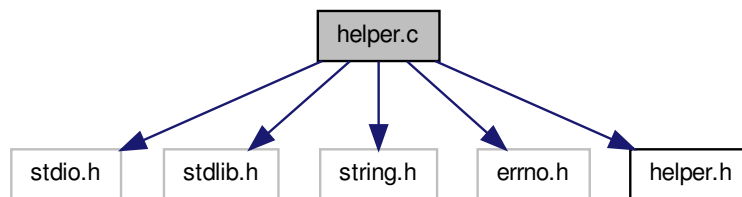
NULL

4.3 helper.c File Reference

Implementation of helper functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include "helper.h"
```

Include dependency graph for helper.c:



Macros

- `#define MAX_BUFFER_SIZE 1024`

Functions

- void `print_errno_message` (const char *message)
Prints an errno error message to stderr.

4.3.1 Detailed Description

Implementation of helper functions.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.3.2 Macro Definition Documentation

4.3.2.1 `#define MAX_BUFFER_SIZE 1024`

4.3.3 Function Documentation

4.3.3.1 void print_errno_message (const char * message)

Prints an errno error message to stderr.

This function accesses the standard global variable `errno` and related function `strerror_r` (a POSIX extension) and can be called after returning from a function which sets `errno` and returns with an error code.

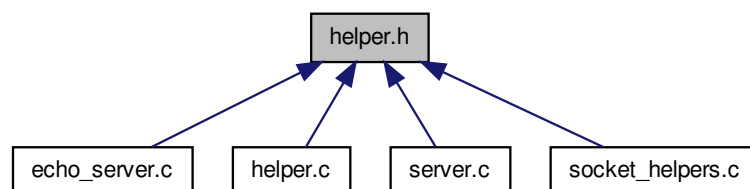
Parameters

<i>message</i>	The error message to show.
----------------	----------------------------

4.4 helper.h File Reference

Interface to helper functions.

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



Macros

- #define `DPRINTF`(arg)
- #define `DFPRINTF`(arg)

Functions

- void `print_errno_message` (const char *message)
Prints an errno error message to stderr.

4.4.1 Detailed Description

Interface to helper functions. Interface to helper functions.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.4.2 Macro Definition Documentation

4.4.2.1 `#define DFPRINTF(arg)`

4.4.2.2 `#define DPRINTF(arg)`

4.4.3 Function Documentation

4.4.3.1 `void print_errno_message (const char * message)`

Prints an errno error message to stderr.

This function accesses the standard global variable `errno` and related function `strerror_r` (a POSIX extension) and can be called after returning from a function which sets `errno` and returns with an error code.

Parameters

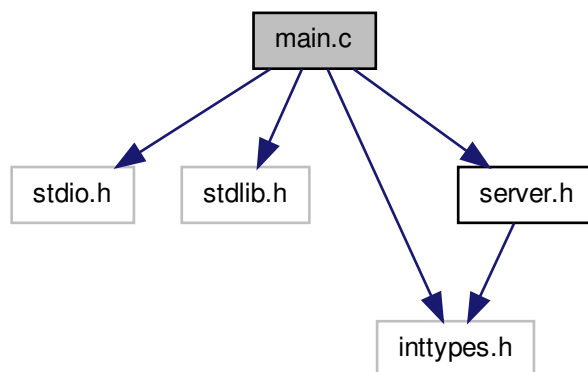
<i>message</i>	The error message to show.
----------------	----------------------------

4.5 main.c File Reference

Main function for echoserver.

```
#include <stdio.h>
#include <stdlib.h>
#include <inttypes.h>
#include "server.h"
```

Include dependency graph for main.c:



Functions

- `uint16_t get_port_from_commandline (const int argc, char **argv)`
Parses the command line for a specified TCP port.
- `int main (int argc, char **argv)`
Main function.

4.5.1 Detailed Description

Main function for echoserver.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.5.2 Function Documentation

4.5.2.1 uint16_t get_port_from_commandline (const int *argc*, char ** *argv*)

Parses the command line for a specified TCP port.

Checks for the existence of a single command line argument, and if one and only one is present, attempts to interpret it as a TCP listening port, between 1 and 49151 (ports above 49151 are ephemeral ports).

Parameters

<i>argc</i>	The number of command line arguments, passed from main()
<i>argv</i>	The command line arguments, passed from main()

Returns

The specified TCP port if successful, or 0 on error.

4.5.2.2 int main (int *argc*, char ** *argv*)

Main function.

Main function.

Returns

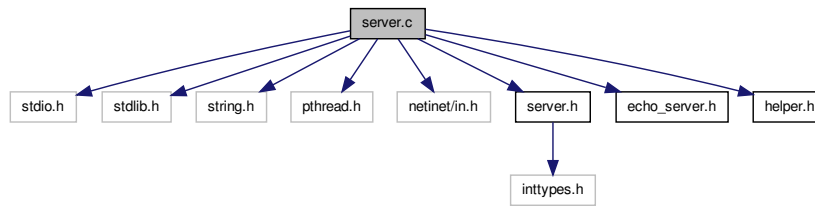
Exit status.

4.6 server.c File Reference

Implementation of listening server functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <netinet/in.h>
#include "server.h"
#include "echo_server.h"
#include "helper.h"
```

Include dependency graph for server.c:



Macros

- #define `IPV6`

Functions

- int `create_server_socket` (const uint16_t listening_port)
- int `start_server` (const int listening_socket)

Starts an active server.

Variables

- static const int `backlog` = 1024

File scope variable for default backlog.

4.6.1 Detailed Description

Implementation of listening server functions.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.6.2 Macro Definition Documentation

4.6.2.1 #define IPV6

4.6.3 Function Documentation

4.6.3.1 int create_server_socket (const uint16_t listening_port)

4.6.3.2 int start_server (const int listening_socket)

Starts an active server.

Connections are passed to a new server thread.

Parameters

<i>listening_socket</i>	A file descriptor for a listening socket.
-------------------------	---

Returns

Returns non-zero on encountering an error. The server runs in an infinite loop, and this function will not return unless an error is countered.

4.6.4 Variable Documentation

4.6.4.1 `const int backlog = 1024` [static]

File scope variable for default backlog.

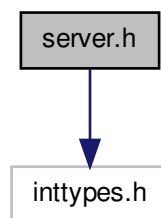
Determines the maximum length to which the queue of pending connections may grow. Used when calling `listen()`.

4.7 server.h File Reference

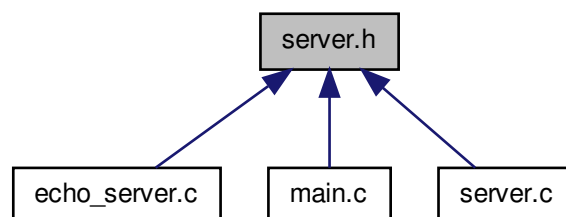
Interface to listening server functions.

```
#include <inttypes.h>
```

Include dependency graph for server.h:



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



Data Structures

- struct [ServerTag](#)

Struct for passing to server threads.

Macros

- #define [DINCREMENT_THREAD_COUNT](#)(arg)
- #define [DDECREMENT_THREAD_COUNT](#)(arg)

Typedefs

- typedef struct [ServerTag](#) [ServerTag](#)

Struct for passing to server threads.

Functions

- int [create_server_socket](#) (const uint16_t listening_port)
- int [start_server](#) (const int listening_socket)

Starts an active server.

4.7.1 Detailed Description

Interface to listening server functions.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.7.2 Macro Definition Documentation

4.7.2.1 #define [DDECREMENT_THREAD_COUNT](#)(*arg*)

4.7.2.2 #define [DINCREMENT_THREAD_COUNT](#)(*arg*)

4.7.3 Typedef Documentation

4.7.3.1 typedef struct [ServerTag](#) [ServerTag](#)

Struct for passing to server threads.

Contains a file descriptor for the connected socket, as the server obviously needs to know this.

4.7.4 Function Documentation

4.7.4.1 `int create_server_socket (const uint16_t listening_port)`

4.7.4.2 `int start_server (const int listening_socket)`

Starts an active server.

Connections are passed to a new server thread.

Parameters

<code>listening_socket</code>	A file descriptor for a listening socket.
-------------------------------	---

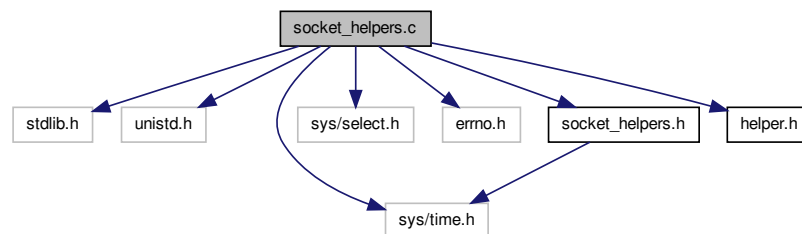
Returns

Returns non-zero on encountering an error. The server runs in an infinite loop, and this function will not return unless an error is countered.

4.8 socket_helpers.c File Reference

Implementation of socket helper functions.

```
#include <stdlib.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/select.h>
#include <errno.h>
#include "socket_helpers.h"
#include "helper.h"
Include dependency graph for socket_helpers.c:
```



Functions

- `ssize_t socket_readline (const int socket, char *buffer, const size_t max_len)`
Reads a ' terminated line from a socket.
- `ssize_t socket_readline_timeout (const int socket, char *buffer, const size_t max_len, struct timeval *time_out)`
Reads a ' terminated line from a socket with timeout.
- `ssize_t socket_writeline (const int socket, const char *buffer, const size_t max_len)`
Writes a line to a socket.

4.8.1 Detailed Description

Implementation of socket helper functions. Implementation of socket helper functions.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.8.2 Function Documentation

4.8.2.1 `ssize_t socket_readline (const int socket, char * buffer, const size_t max_len)`

Reads a '

' terminated line from a socket.

The function will not overwrite the buffer, so `max_len` should be the size of the whole buffer, and function will at most write `max_len - 1` characters plus the terminating `"\0"`.

Parameters

<i>socket</i>	File description of the socket
<i>buffer</i>	The buffer into which to read
<i>max_len</i>	The maximum number of characters to read, including the terminating <code>"\0"</code> .

Returns

The number of characters read, or -1 on encountering an error.

4.8.2.2 `ssize_t socket_readline_timeout (const int socket, char * buffer, const size_t max_len, struct timeval * time_out)`

Reads a '

' terminated line from a socket with timeout.

Behaves the same as [socket_readline\(\)](#), except it will time out if no input is available on the socket after the specified time.

Parameters

<i>socket</i>	File description of the socket
<i>buffer</i>	The buffer into which to read
<i>max_len</i>	The maximum number of characters to read, including the terminating <code>"\0"</code> . \param <i>time_out</i> A pointer to a <code>timeval</code> struct containing the timeout period. Note that some implementations of <code>select()</code> may alter this variable, so the calling function should consider it unusable after return. In addition, on such an implementation, the value will specify the cumulative timeout period over the entire read line operation, rather than resetting after reading each character.

Returns

The number of characters read, or -1 on encountering an error.

4.8.2.3 ssize_t socket_writeline (const int *socket*, const char * *buffer*, const size_t *max_len*)

Writes a line to a socket.

Parameters

<i>socket</i>	File description of the socket
<i>buffer</i>	The buffer from which to write.
<i>max_len</i>	The maximum number of characters to read from the buffer.

Returns

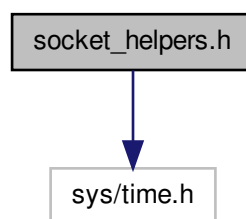
The number of characters written, or -1 on encountering an error.

4.9 socket_helpers.h File Reference

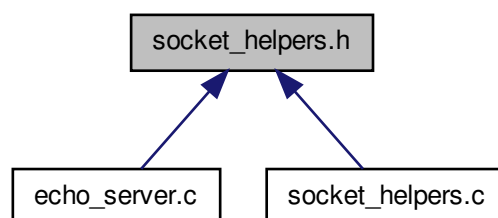
Interface to socket helper functions.

```
#include <sys/time.h>
```

Include dependency graph for socket_helpers.h:



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



Functions

- ssize_t [socket_readline](#) (const int *l_socket*, char **buffer*, const size_t *max_len*)

- Reads a ' terminated line from a socket.*
- ssize_t [socket_readline_timeout](#) (const int l_socket, char *buffer, const size_t max_len, struct timeval *time_out)
- Reads a ' terminated line from a socket with timeout.*
- ssize_t [socket_writeline](#) (const int l_socket, const char *buffer, const size_t max_len)
- Writes a line to a socket.*

4.9.1 Detailed Description

Interface to socket helper functions.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.9.2 Function Documentation

4.9.2.1 ssize_t socket_readline (const int socket, char * buffer, const size_t max_len)

Reads a ' terminated line from a socket.

' terminated line from a socket.

The function will not overwrite the buffer, so max_len should be the size of the whole buffer, and function will at most write max_len - 1 characters plus the terminating "\0".

Parameters

<i>socket</i>	File description of the socket
<i>buffer</i>	The buffer into which to read
<i>max_len</i>	The maximum number of characters to read, including the terminating "\0".

Returns

The number of characters read, or -1 on encountering an error.

4.9.2.2 ssize_t socket_readline_timeout (const int socket, char * buffer, const size_t max_len, struct timeval * time_out)

Reads a ' terminated line from a socket with timeout.

' terminated line from a socket with timeout.

Behaves the same as [socket_readline\(\)](#), except it will time out if no input is available on the socket after the specified time.

Parameters

<i>socket</i>	File description of the socket
<i>buffer</i>	The buffer into which to read

<i>max_len</i>	The maximum number of characters to read, including the terminating “\0”. <i>time_out</i> A pointer to a <code>timeval</code> struct containing the timeout period. Note that some implementations of <code>select()</code> may alter this variable, so the calling function should consider it unusable after return. In addition, on such an implementation, the value will specify the cumulative timeout period over the entire read line operation, rather than resetting after reading each character.
----------------	---

Returns

The number of characters read, or -1 on encountering an error.

4.9.2.3 `ssize_t socket_writeline (const int socket, const char * buffer, const size_t max_len)`

Writes a line to a socket.

Parameters

<i>socket</i>	File description of the socket
<i>buffer</i>	The buffer from which to write.
<i>max_len</i>	The maximum number of characters to read from the buffer.

Returns

The number of characters written, or -1 on encountering an error.

Index

- backlog
 - server.c, [15](#)
- c_socket
 - ServerTag, [5](#)
- create_server_socket
 - server.c, [14](#)
 - server.h, [17](#)
- DFPRINTF
 - helper.h, [12](#)
- DPRINTF
 - helper.h, [12](#)
- echo_server
 - echo_server.c, [8](#)
 - echo_server.h, [9](#)
- echo_server.c, [7](#)
 - echo_server, [8](#)
 - MAX_BUFFER_LEN, [8](#)
 - time_out_msg, [8](#)
 - time_out_secs, [8](#)
 - time_out_usecs, [8](#)
- echo_server.h, [9](#)
 - echo_server, [9](#)
- get_port_from_commandline
 - main.c, [13](#)
- helper.c, [10](#)
 - MAX_BUFFER_SIZE, [10](#)
 - print_errno_message, [10](#)
- helper.h, [11](#)
 - DFPRINTF, [12](#)
 - DPRINTF, [12](#)
 - print_errno_message, [12](#)
- IPV6
 - server.c, [14](#)
- MAX_BUFFER_LEN
 - echo_server.c, [8](#)
- MAX_BUFFER_SIZE
 - helper.c, [10](#)
- main
 - main.c, [13](#)
- main.c, [12](#)
 - get_port_from_commandline, [13](#)
 - main, [13](#)
- print_errno_message
 - helper.c, [10](#)
 - helper.h, [12](#)
- server.c, [13](#)
 - backlog, [15](#)
 - create_server_socket, [14](#)
 - IPV6, [14](#)
 - start_server, [14](#)
- server.h, [15](#)
 - create_server_socket, [17](#)
 - ServerTag, [16](#)
 - start_server, [17](#)
- ServerTag, [5](#)
 - c_socket, [5](#)
 - server.h, [16](#)
- socket_helpers.c, [17](#)
 - socket_readline, [18](#)
 - socket_readline_timeout, [18](#)
 - socket_writeline, [18](#)
- socket_helpers.h, [19](#)
 - socket_readline, [20](#)
 - socket_readline_timeout, [20](#)
 - socket_writeline, [21](#)
- socket_readline
 - socket_helpers.c, [18](#)
 - socket_helpers.h, [20](#)
- socket_readline_timeout
 - socket_helpers.c, [18](#)
 - socket_helpers.h, [20](#)
- socket_writeline
 - socket_helpers.c, [18](#)
 - socket_helpers.h, [21](#)
- start_server
 - server.c, [14](#)
 - server.h, [17](#)
- time_out_msg
 - echo_server.c, [8](#)
- time_out_secs
 - echo_server.c, [8](#)
- time_out_usecs
 - echo_server.c, [8](#)