## sockethelpers

Generated by Doxygen 1.8.1.2

Sat Aug 31 2013 21:40:16

## **Contents**

1	Data	Structi	ıre Index	(	1
	1.1	Data S	tructures		 1
2	File	Index			3
	2.1	File Lis	st		 3
3	Data	Structi	ıre Docur	mentation	5
	3.1	Server	Tag Struct	t Reference	 5
		3.1.1	Detailed	Description	 5
		3.1.2	Field Do	ocumentation	 5
			3.1.2.1	c_socket	 5
4	File	Docume	entation		7
	4.1	socket	_helpers.h	h File Reference	 7
		4.1.1	Detailed	Description	 8
	4.2	socket	_helpers_i	_main.c File Reference	 8
		4.2.1	Detailed	I Description	 9
		4.2.2	Function	Documentation	 9
			4.2.2.1	conn_socket_from_string	 9
			4.2.2.2	ignore_sigpipe	 10
			4.2.2.3	port_from_string	 10
			4.2.2.4	socket_readline	 10
			4.2.2.5	socket_readline_timeout	 10
			4.2.2.6	socket_writeline	 11
	4.3	socket	_helpers_ı	main.h File Reference	 11
		4.3.1	Detailed	Description	 12
		4.3.2	Function	Documentation	 12
			4.3.2.1	conn_socket_from_string	 12
			4.3.2.2	ignore_sigpipe	 13
			4.3.2.3	port_from_string	 13
			4.3.2.4	socket_readline	13
			4325	socket readline timeout	13

ii CONTENTS

		4.3.2.6	socket_wri	teline .				 		 			 		14
4.4	socket	_helpers_s	server.c File	Referen	ce .			 		 			 		14
	4.4.1	Detailed	Description					 		 			 		15
	4.4.2	Function	Documenta	tion				 		 			 		15
		4.4.2.1	create_tcp	_server_	_socke	t		 		 			 		15
		4.4.2.2	start_threa	ded_tcp	_serve	r		 		 			 		15
4.5	socket	_helpers_s	server.h File	Referen	ce .			 		 			 		15
	4.5.1	Detailed	Description					 		 			 		17
	4.5.2	Typedef I	Documentati	on				 		 			 		17
		4.5.2.1	ServerTag					 		 			 		17
	4.5.3	Function	Documenta	tion				 		 			 		17
		4.5.3.1	create_tcp	_server_	socke	t		 		 			 		17
		4532	start threa	ded ton	serve	r									17

# **Data Structure Index**

1.1	1 Data Structures	
Here	ere are the data structures with brief descriptions:	

ServerTag															
Struct for passing to server threads										 					5

2 Data Structure Index

# File Index

## 2.1 File List

Here is a list of all documented files with brief descriptions:

7
8
11
14
15
1

File Index

## **Data Structure Documentation**

## 3.1 ServerTag Struct Reference

Struct for passing to server threads.

```
#include <socket_helpers_server.h>
```

#### **Data Fields**

• int c\_socket

#### 3.1.1 Detailed Description

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:

• socket\_helpers\_server.h

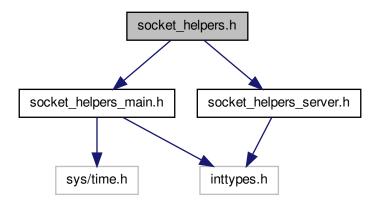


## **File Documentation**

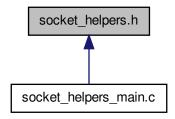
## 4.1 socket\_helpers.h File Reference

Interface to socket helper library.

```
#include "socket_helpers_main.h"
#include "socket_helpers_server.h"
Include dependency graph for socket_helpers.h:
```



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



### 4.1.1 Detailed Description

Interface to socket helper library.

**Author** 

Paul Griffiths

#### Copyright

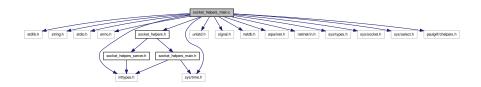
Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. http-://www.gnu.org/licenses/

## 4.2 socket\_helpers\_main.c File Reference

Implementation of main socket helper functions.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <errno.h>
#include <inttypes.h>
#include <unistd.h>
#include <signal.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/select.h>
#include <paulgrif/chelpers.h>
#include "socket_helpers.h"
```

Include dependency graph for socket\_helpers\_main.c:



#### **Macros**

• #define MAX BUFFER SIZE 1024

Maximum character buffer size.

#### **Functions**

• ssize\_t socket\_readline (const int socket, char \*buffer, const size\_t max\_len)

Reads an  $\r n$  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 an  $\r n$  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.

uint16\_t port\_from\_string (const char \*port\_str)

Extracts a valid TCP/UDP port from a string.

int conn\_socket\_from\_string (const char \*host, const char \*port)

Creates a connected sock from a hostname and port.

void ignore\_sigpipe (void)

Ignores the SIGPIPE signal.

#### 4.2.1 Detailed Description

#### 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 int conn\_socket\_from\_string ( const char \* host, const char \* port )

#### **Parameters**

host	A string containing the hostname to which to connect.
port	A string containing the port to which to connect.

#### Returns

The file descriptor of the connected socket on success, or -1 on failure.

#### 4.2.2.2 void ignore\_sigpipe (void)

The write() system call will, when writing to a closed socket, elicit an RST (reset) flag. A second write() system call will trigger a SIGPIPE signal to be raised. The default action of SIGPIPE is to terminate the program, with no error message, which is not desirable. If we want to do anything special when SIGPIPE is triggered, we could set up a handler, but if we don't, then ignoring SIGPIPE is fine, provided our socket functions respond appropriately to the condition (write() will return EPIPE after an ignored SIGPIPE signal).

4.2.2.3 uint16\_t port\_from\_string ( const char \* port\_str )

#### **Parameters**

port_str   The string from which to extract
---

#### Returns

The port number on success, or zero if port\_str does not contain a valid TCP/UDP port (port 0 is reserved and cannot be used).

4.2.2.4 ssize\_t socket\_readline ( const int socket, char \* buffer, const size\_t max\_len )

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  $\0$ . Any terminating CR or LF characters will be stripped.

#### **Parameters**

socket	File description of the socket
buffer	The buffer into which to read
max_len	The maximum number of characters to read, including the terminating $\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$

#### Returns

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

4.2.2.5 ssize\_t socket\_readline\_timeout ( const int socket, char \* buffer, const size\_t max\_len, struct timeval \* time\_out )

Behaves the same as socket\_readline(), except it will time out if no input is available on the socket after the specified time. Any terminating CR or LF characters will be stripped.

#### **Parameters**

socket	File description of the socket
buffer	The buffer into which to read
max_len	The maximum number of characters to read, including the terminating \0.
time_out	A pointer to a timeval struct containing the timeout period. Note that some implementations
	of select () 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.

4.2.2.6 ssize\_t socket\_writeline ( const int socket, const char \* buffer, const size\_t max\_len )

The function adds a network-standard terminating CRLF, so the provided string should not normally end in any newline characters.

#### **Parameters**

socket	File description of the socket
buffer	The buffer from which to write.
max_len	The maximum number of characters to write to the buffer. Due to the addition of CRLF, max-
	_len + 2 characters may actually be written.

#### Returns

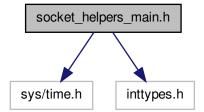
The number of characters written, or -1 on encountering an error.

## 4.3 socket\_helpers\_main.h File Reference

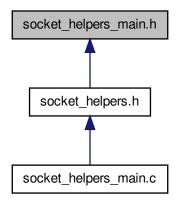
Interface to main socket helper functions.

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

Include dependency graph for socket\_helpers\_main.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 an  $\r \n$  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 an  $\r n$  terminated line from a socket with timeout.

• ssize\_t socket\_writeline (const int I\_socket, const char \*buffer, const size\_t max\_len)

Writes a line to a socket.

uint16\_t port\_from\_string (const char \*port\_str)

Extracts a valid TCP/UDP port from a string.

int conn\_socket\_from\_string (const char \*host, const char \*port)

Creates a connected sock from a hostname and port.

• void ignore\_sigpipe (void)

Ignores the SIGPIPE signal.

### 4.3.1 Detailed Description

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 Function Documentation

4.3.2.1 int conn\_socket\_from\_string ( const char \* host, const char \* port )

#### **Parameters**

host	A string containing the hostname to which to connect.
port	A string containing the port to which to connect.

#### **Returns**

The file descriptor of the connected socket on success, or -1 on failure.

#### 4.3.2.2 void ignore\_sigpipe (void)

The write() system call will, when writing to a closed socket, elicit an RST (reset) flag. A second write() system call will trigger a SIGPIPE signal to be raised. The default action of SIGPIPE is to terminate the program, with no error message, which is not desirable. If we want to do anything special when SIGPIPE is triggered, we could set up a handler, but if we don't, then ignoring SIGPIPE is fine, provided our socket functions respond appropriately to the condition (write() will return EPIPE after an ignored SIGPIPE signal).

4.3.2.3 uint16\_t port\_from\_string ( const char \* port\_str )

#### **Parameters**

port_str	The string from which to extract

#### Returns

The port number on success, or zero if port\_str does not contain a valid TCP/UDP port (port 0 is reserved and cannot be used).

4.3.2.4 ssize\_t socket\_readline ( const int socket, char \* buffer, const size\_t max\_len )

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  $\0$ . Any terminating CR or LF characters will be stripped.

#### Parameters

socket	File description of the socket
buffer	The buffer into which to read
max_len	The maximum number of characters to read, including the terminating $\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$

#### Returns

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

4.3.2.5 ssize\_t socket\_readline\_timeout ( const int socket, char \* buffer, const size\_t max\_len, struct timeval \* time\_out )

Behaves the same as socket\_readline(), except it will time out if no input is available on the socket after the specified time. Any terminating CR or LF characters will be stripped.

#### **Parameters**

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

time_out	A pointer to a timeval struct containing the timeout period. Note that some implementations
	of select () 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.

4.3.2.6 ssize\_t socket\_writeline ( const int socket, const char \* buffer, const size\_t max\_len )

The function adds a network-standard terminating CRLF, so the provided string should not normally end in any newline characters.

#### **Parameters**

socket	File description of the socket
buffer	The buffer from which to write.
max_len	The maximum number of characters to write to the buffer. Due to the addition of CRLF, max-
	_len + 2 characters may actually be written.

#### Returns

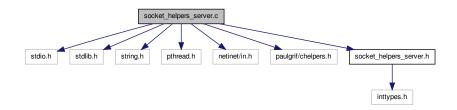
The number of characters written, or -1 on encountering an error.

## 4.4 socket\_helpers\_server.c File Reference

Implementation of server helper functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <netinet/in.h>
#include <paulgrif/chelpers.h>
#include "socket_helpers_server.h"
```

Include dependency graph for socket\_helpers\_server.c:



#### **Macros**

• #define IPV6

Create an IPv6 rather than IPv4 listening socket.

#### **Functions**

int create\_tcp\_server\_socket (const uint16\_t listening\_port)

Creates a TCP listening socket.

int start threaded tcp server (const int listening socket, void \*(\*sfunc)(void \*))

Starts an active server.

#### 4.4.1 Detailed Description

Implementation of server 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 Function Documentation

4.4.2.1 int create\_tcp\_server\_socket ( const uint16\_t listening\_port )

The function creates an IPv4 socket by default, but creates an IPv6 socket if the IPV6 preprocessor macro is defined.

#### **Parameters**

listening_port	The port the socket should listen on

#### **Returns**

The file descriptor of the created listening socket on success, or -1 on encountering an error.

4.4.2.2 int start\_threaded\_tcp\_server ( const int listening\_socket, void \*(\*)(void \*) sfunc )

Connections are passed to a new server thread.

#### **Parameters**

listening_socket	A file descriptor for a listening socket.
sfunc	A pointer to a server thread function. The function should return a pointer to void and accept a
	single pointer to void as an argument, which should be interpreted as a pointer to a Server-
	Tag struct.

#### Returns

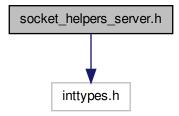
Returns non-zero on encountering an error. The server runs in an infinite loop, and this function will not return unless an error is countered.

### 4.5 socket\_helpers\_server.h File Reference

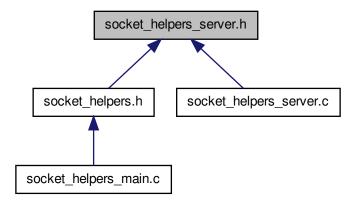
Interface to server helper functions.

#include <inttypes.h>

Include dependency graph for socket\_helpers\_server.h:



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



#### **Data Structures**

• struct ServerTag

Struct for passing to server threads.

#### **Typedefs**

• typedef struct ServerTag ServerTag Struct for passing to server threads.

#### **Functions**

• int create\_tcp\_server\_socket (const uint16\_t listening\_port)

Creates a TCP listening socket.

int start\_threaded\_tcp\_server (const int listening\_socket, void \*(\*sfunc)(void \*))
 Starts an active server.

### 4.5.1 Detailed Description

Interface to server 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/

#### 4.5.2 Typedef Documentation

#### 4.5.2.1 typedef struct ServerTag ServerTag

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

#### 4.5.3 Function Documentation

4.5.3.1 int create\_tcp\_server\_socket ( const uint16\_t listening\_port )

The function creates an IPv4 socket by default, but creates an IPv6 socket if the IPV6 preprocessor macro is defined.

### Parameters

listening_port	The port the socket should listen on

#### Returns

The file descriptor of the created listening socket on success, or -1 on encountering an error.

4.5.3.2 int start\_threaded\_tcp\_server ( const int listening\_socket, void \*(\*)(void \*) sfunc )

Connections are passed to a new server thread.

#### **Parameters**

listening_socket	A file descriptor for a listening socket.
sfunc	A pointer to a server thread function. The function should return a pointer to void and accept a
	single pointer to void as an argument, which should be interpreted as a pointer to a Server-
	Tag struct.

#### 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.

## Index

c_socket
ServerTag, 5
conn_socket_from_string
socket_helpers_main.c, 9
socket_helpers_main.h, 12
create_tcp_server_socket
socket_helpers_server.c, 15
socket_helpers_server.h, 17
ignore_sigpipe
socket_helpers_main.c, 10
socket_helpers_main.h, 13
_ , _
port_from_string
socket_helpers_main.c, 10
socket_helpers_main.h, 13
ServerTag, 5
c socket, 5
socket_helpers_server.h, 17
socket helpers.h, 7
socket helpers main.c, 8
conn socket from string, 9
ignore_sigpipe, 10
port from string, 10
socket readline, 10
socket readline timeout, 10
socket writeline, 11
socket_helpers_main.h, 11
conn_socket_from_string, 12
ignore_sigpipe, 13
port_from_string, 13
socket_readline, 13
socket_readline_timeout, 13
socket_writeline, 14
socket_helpers_server.c, 14
create_tcp_server_socket, 15
start_threaded_tcp_server, 15
socket_helpers_server.h, 15
create_tcp_server_socket, 17
ServerTag, 17
start_threaded_tcp_server, 17
socket_readline
socket_helpers_main.c, 10
socket_helpers_main.h, 13
socket_readline_timeout
socket_helpers_main.c, 10
socket_helpers_main.h, 13
socket_writeline
socket_helpers_main.c, 11

socket\_helpers\_main.h, 14 start\_threaded\_tcp\_server socket\_helpers\_server.c, 15 socket\_helpers\_server.h, 17