sockethelpers

Generated by Doxygen 1.8.1.2

Sat Aug 31 2013 20:54:11

Contents

Chapter 1

File Index

1.1 File List

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

Chapter 2

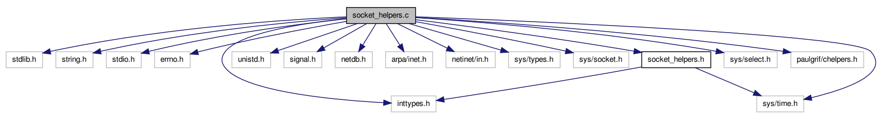
File Documentation

2.1 socket_helpers.c File Reference

Implementation of 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.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.

2.1.1 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/`

2.1.2 Function Documentation

2.1.2.1 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.

2.1.2.2 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).

2.1.2.3 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).

2.1.2.4 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 `\0`.

Returns

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

2.1.2.5 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.

2.1.2.6 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

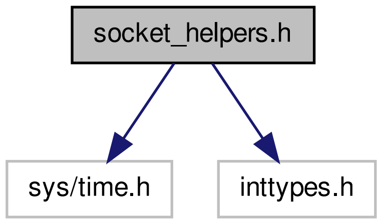
The number of characters written, or -1 on encountering an error.

2.2 socket_helpers.h File Reference

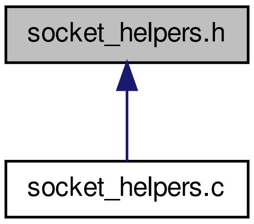
Interface to socket helper functions.

```
#include <sys/time.h>
#include <inttypes.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 an $\backslash r \backslash n$ *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 $\backslash r \backslash n$ *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.

- 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.

2.2.1 Detailed Description

Author

Paul Griffiths

Copyright

2.2.2 Function Documentation

2.2.2.1 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.

2.2.2.2 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).

2.2.2.3 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).

2.2.2.4 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 `\0`.

Returns

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

2.2.2.5 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.

2.2.2.6 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.

Index