NAME

explain_socket − explain socket(2) errors

SYNOPSIS

#include <libexplain/socket.h>

const char *explain_socket(int domain, int type, int protocol);
const char *explain_errno_socket(int errnum, int domain, int type, int protocol);
void explain_message_socket(char *message, int message_size, int domain, int type, int protocol);
void explain_message_errno_socket(char *message, int message_size, int errnum, int domain, int type, int protocol);

DESCRIPTION

These functions may be used to obtain explanations for errors returned by the *socket*(2) system call.

explain_socket

const char *explain_socket(int domain, int type, int protocol);

The **explain_socket** function is used to obtain an explanation of an error returned by the *socket*(2) system call. The least the message will contain is the value of `strerror(errno)`, but usually it will do much better, and indicate the underlying cause in more detail.

The *errno* global variable will be used to obtain the error value to be decoded.

This function is intended to be used in a fashion similar to the following example:

```
if (socket(domain, type, protocol) < 0)
{
    fprintf(stderr, "%s\n", explain_socket(domain, type, protocol));
    exit(EXIT_FAILURE);
}
```

domain The original domain, exactly as passed to the *socket*(2) system call.

type The original type, exactly as passed to the *socket*(2) system call.

protocol The original protocol, exactly as passed to the *socket*(2) system call.

Returns: The message explaining the error. This message buffer is shared by all libexplain functions which do not supply a buffer in their argument list. This will be overwritten by the next call to any libexplain function which shares this buffer, including other threads.

Note: This function is **not** thread safe, because it shares a return buffer across all threads, and many other functions in this library.

explain_errno_socket

const char *explain_errno_socket(int errnum, int domain, int type, int protocol);

The **explain_errno_socket** function is used to obtain an explanation of an error returned by the *socket*(2) system call. The least the message will contain is the value of `strerror(errnum)`, but usually it will do much better, and indicate the underlying cause in more detail.

This function is intended to be used in a fashion similar to the following example:

```
if (socket(domain, type, protocol) < 0)
{
    int err = errno;
    fprintf(stderr, "%s\n", explain_errno_socket(err,
        domain, type, protocol));
    exit(EXIT_FAILURE);
}
```

errnum The error value to be decoded, usually obtained from the *errno* global variable just before this function is called. This is necessary if you need to call **any** code between the system call to be explained and this function, because many libc functions will alter the value of *errno*.

domain The original domain, exactly as passed to the *socket*(2) system call.

type The original type, exactly as passed to the *socket*(2) system call.

protocol The original protocol, exactly as passed to the *socket*(2) system call.

Returns: The message explaining the error. This message buffer is shared by all libexplain functions which do not supply a buffer in their argument list. This will be overwritten by the next call to any libexplain function which shares this buffer, including other threads.

Note: This function is **not** thread safe, because it shares a return buffer across all threads, and many other functions in this library.

explain_message_socket

void explain_message_socket(char *message, int message_size, int domain, int type, int protocol);

The **explain_message_socket** function may be used to obtain an explanation of an error returned by the *socket*(2) system call. The least the message will contain is the value of `strerror(errno)`, but usually it will do much better, and indicate the underlying cause in more detail.

The *errno* global variable will be used to obtain the error value to be decoded.

This function is intended to be used in a fashion similar to the following example:

```
if (socket(domain, type, protocol) < 0)
{
    char message[3000];
    explain_message_socket(message, sizeof(message), domain, type, protocol
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

message The location in which to store the returned message. If a suitable message return buffer is supplied, this function is thread safe.

message_size
The size in bytes of the location in which to store the returned message.

domain The original domain, exactly as passed to the *socket*(2) system call.

type The original type, exactly as passed to the *socket*(2) system call.

protocol The original protocol, exactly as passed to the *socket*(2) system call.

explain_message_errno_socket

void explain_message_errno_socket(char *message, int message_size, int errnum, int domain, int type, int protocol);

The **explain_message_errno_socket** function may be used to obtain an explanation of an error returned by the *socket*(2) system call. The least the message will contain is the value of `strerror(errnum)`, but usually it will do much better, and indicate the underlying cause in more detail.

This function is intended to be used in a fashion similar to the following example:

```
if (socket(domain, type, protocol) < 0)
{
    int err = errno;
    char message[3000];
    explain_message_errno_socket(message, sizeof(message), err,
        domain, type, protocol);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

message The location in which to store the returned message. If a suitable message return buffer is supplied, this function is thread safe.

message_size
The size in bytes of the location in which to store the returned message.

errnum The error value to be decoded, usually obtained from the *errno* global variable just before this function is called. This is necessary if you need to call **any** code between the system call to be explained and this function, because many libc functions will alter the value of *errno*.

domain The original domain, exactly as passed to the *socket*(2) system call.

type The original type, exactly as passed to the *socket*(2) system call.

protocol The original protocol, exactly as passed to the *socket*(2) system call.

SEE ALSO

socket(2)
create an endpoint for communication

explain_socket_or_die(3)
create an endpoint for communication and report errors

COPYRIGHT