NAME

explain_socketpair − explain *socketpair*(2) errors

SYNOPSIS

#include <libexplain/socketpair.h>

const char *explain_socketpair(int domain, int type, int protocol, int *sv);
const char *explain_errno_socketpair(int errnum, int domain, int type, int protocol, int *sv);
void explain_message_socketpair(char *message, int message_size, int domain, int type, int protocol, int *sv);
void explain_message_errno_socketpair(char *message, int message_size, int errnum, int domain, int type, int protocol, int *sv);

DESCRIPTION

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

explain_socketpair

const char *explain_socketpair(int domain, int type, int protocol, int *sv);

The **explain_socketpair** function is used to obtain an explanation of an error returned by the *socketpair*(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.

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

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

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

sv The original sv, exactly as passed to the *socketpair*(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.

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

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

The above code example is available pre-packaged as the *explain_socketpair_or_die*(3) function.

explain_errno_socketpair

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

The **explain_errno_socketpair** function is used to obtain an explanation of an error returned by the *socketpair*(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.

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 *socketpair*(2) system call.

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

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

sv The original sv, exactly as passed to the *socketpair*(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.

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

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

The above code example is available pre-packaged as the *explain_socketpair_or_die*(3) function.

explain_message_socketpair

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

The **explain_message_socketpair** function is used to obtain an explanation of an error returned by the *socketpair*(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.

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 *socketpair*(2) system call.

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

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

sv The original sv, exactly as passed to the *socketpair*(2) system call.

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

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

The above code example is available pre-packaged as the *explain_socketpair_or_die*(3) function.

explain_message_errno_socketpair

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

The **explain_message_errno_socketpair** function is used to obtain an explanation of an error returned by the *socketpair*(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.

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 *socketpair*(2) system call.

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

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

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

The above code example is available pre-packaged as the *explain_socketpair_or_die*(3) function.

SEE ALSO

socketpair(2)
create a pair of connected sockets

explain_socketpair_or_die(3)
create a pair of connected sockets and report errors

COPYRIGHT