NAME

explain_pipe2 − explain *pipe2*(2) errors

SYNOPSIS

#include <libexplain/pipe2.h>

const char *explain_pipe2(int *fildes, int flags);
const char *explain_errno_pipe2(int errnum, int *fildes, int flags);
void explain_message_pipe2(char *message, int message_size, int *fildes, int flags);
void explain_message_errno_pipe2(char *message, int message_size, int errnum, int *fildes, int flags);

DESCRIPTION

These functions may be used to obtain explanations for errors returned by the *pipe2*(2) system call.

explain_pipe2

const char *explain_pipe2(int *fildes, int flags);

The **explain_pipe2** function is used to obtain an explanation of an error returned by the *pipe2*(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.

fildes The original fildes, exactly as passed to the *pipe2*(2) system call.

flags The original flags, exactly as passed to the *pipe2*(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 (pipe2(fildes, flags) < 0)
{
    fprintf(stderr, "%s\n", explain_pipe2(fildes, flags));
     exit(EXIT_FAILURE);
}
```

The above code example is available pre−packaged as the *explain_pipe2_or_die*(3) function.

explain_errno_pipe2

const char *explain_errno_pipe2(int errnum, int *fildes, int flags);

The **explain_errno_pipe2** function is used to obtain an explanation of an error returned by the *pipe2*(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*.

fildes The original fildes, exactly as passed to the *pipe2*(2) system call.

flags The original flags, exactly as passed to the *pipe2*(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 (pipe2(fildes, flags) < 0)
{
     int err = errno;
    fprintf(stderr, "%s\n", explain_errno_pipe2(err, fildes,
    flags));
```

```
        exit(EXIT_FAILURE);
    }
```

The above code example is available pre−packaged as the *explain_pipe2_or_die*(3) function.

explain_message_pipe2

void explain_message_pipe2(char *message, int message_size, int *fildes, int flags);

The **explain_message_pipe2** function is used to obtain an explanation of an error returned by the *pipe2*(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.

fildes The original fildes, exactly as passed to the *pipe2*(2) system call.

flags The original flags, exactly as passed to the *pipe2*(2) system call.

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

```
if (pipe2(fildes, flags) < 0)
{
    char message[3000];
   explain_message_pipe2(message, sizeof(message), fildes,
   flags);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

The above code example is available pre−packaged as the *explain_pipe2_or_die*(3) function.

explain_message_errno_pipe2

void explain_message_errno_pipe2(char *message, int message_size, int errnum, int *fildes, int flags);

The **explain_message_errno_pipe2** function is used to obtain an explanation of an error returned by the *pipe2*(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*.

fildes The original fildes, exactly as passed to the *pipe2*(2) system call.

flags The original flags, exactly as passed to the *pipe2*(2) system call.

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

```
if (pipe2(fildes, flags) < 0)
{
    int err = errno;
    char message[3000];
   explain_message_errno_pipe2(message, sizeof(message), err,
   fildes, flags);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

The above code example is available pre−packaged as the *explain_pipe2_or_die*(3) function.

SEE ALSO

pipe2(2)	create pipe
explain_pipe2_or_die(3)	create pipe and report errors

COPYRIGHT