NAME

explain_fgets − explain fgets(3) errors

SYNOPSIS

#include <libexplain/fgets.h>

const char *explain_fgets(char *data, int data_size, FILE *fp);
const char *explain_errno_fgets(int errnum, char *data, int data_size, FILE *fp);
void explain_message_fgets(char *message, int message_size, char *data, int data_size, FILE *fp);
void explain_message_errno_fgets(char *message, int message_size, int errnum, char *data, int data_size, FILE *fp);

DESCRIPTION

These functions may be used to obtain explanations for errors returned by the *fgets*(3) system call.

explain_fgets

const char *explain_fgets(char *data, int data_size, FILE *fp);

The **explain_fgets** function is used to obtain an explanation of an error returned by the *fgets*(3) 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 (fgets(data, data_size, fp) < 0)
{
    fprintf(stderr, "%s\n", explain_fgets(data, data_size, fp));
    exit(EXIT_FAILURE);
}
```

data The original data, exactly as passed to the *fgets*(3) system call.

data_size
The original data_size, exactly as passed to the *fgets*(3) system call.

fp The original fp, exactly as passed to the *fgets*(3) 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_fgets

const char *explain_errno_fgets(int errnum, char *data, int data_size, FILE *fp);

The **explain_errno_fgets** function is used to obtain an explanation of an error returned by the *fgets*(3) 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 (fgets(data, data_size, fp) < 0)
{
    int err = errno;
    fprintf(stderr, "%s\n", explain_errno_fgets(err, data, data_size, fp));
    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*.

data The original data, exactly as passed to the *fgets*(3) system call.

data_size
The original data_size, exactly as passed to the *fgets*(3) system call.

fp The original fp, exactly as passed to the *fgets*(3) 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_fgets

void explain_message_fgets(char *message, int message_size, char *data, int data_size, FILE *fp);

The **explain_message_fgets** function may be used to obtain an explanation of an error returned by the *fgets*(3) 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 (fgets(data, data_size, fp) < 0)
{
    char message[3000];
    explain_message_fgets(message, sizeof(message), data, data_size, fp);
    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.

data The original data, exactly as passed to the *fgets*(3) system call.

data_size
The original data_size, exactly as passed to the *fgets*(3) system call.

fp The original fp, exactly as passed to the *fgets*(3) system call.

explain_message_errno_fgets

void explain_message_errno_fgets(char *message, int message_size, int errnum, char *data, int data_size, FILE *fp);

The **explain_message_errno_fgets** function may be used to obtain an explanation of an error returned by the *fgets*(3) 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 (fgets(data, data_size, fp) < 0)
{
    int err = errno;
    char message[3000];
    explain_message_errno_fgets(message, sizeof(message), err,
        data, data_size, fp);
    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*.

data The original data, exactly as passed to the *fgets*(3) system call.

data_size
The original data_size, exactly as passed to the *fgets*(3) system call.

fp The original fp, exactly as passed to the *fgets*(3) system call.

SEE ALSO

fgets(3) input of strings

explain_fgets_or_die(3)
input of strings and report errors

COPYRIGHT