

explain_setbuf(3)

explain_setbuf(3)

NAMEexplain_setbuf – explain *setbuf*(3) errors**SYNOPSIS**

```
#include <libexplain/setbuf.h>

const char *explain_setbuf(FILE *fp, char *data);
const char *explain_errno_setbuf(int errnum, FILE *fp, char *data);
void explain_message_setbuf(char *message, int message_size, FILE *fp, char *data);
void explain_message_errno_setbuf(char *message, int message_size, int errnum, FILE *fp, char *data);
```

DESCRIPTION

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

explain_setbuf

```
const char *explain_setbuf(FILE *fp, char *data);
```

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

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

data The original data, exactly as passed to the *setbuf*(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.

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

```
errno = 0;
void result = setbuf(fp, data);
if (result < 0 && errno != 0)
{
    fprintf(stderr, "%s\n", explain_setbuf(fp, data));
    exit(EXIT_FAILURE);
}
```

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

explain_errno_setbuf

```
const char *explain_errno_setbuf(int errnum, FILE *fp, char *data);
```

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

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*.

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

data The original data, exactly as passed to the *setbuf*(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.

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

```
errno = 0;
void result = setbuf(fp, data);
```



explain_setbuf(3)

explain_setbuf(3)

```

if (result < 0 && errno != 0)
{
    int err = errno;
    fprintf(stderr, "%s\n", explain_errno_setbuf(err, fp,
    data));
    exit(EXIT_FAILURE);
}

```

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

explain_message_setbuf

```
void explain_message_setbuf(char *message, int message_size, FILE *fp, char *data);
```

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

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.

fp

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

data

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

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

```

errno = 0;
void result = setbuf(fp, data);
if (result < 0 && errno != 0)
{
    char message[3000];
    explain_message_setbuf(message, sizeof(message), fp, data);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}

```

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

explain_message_errno_setbuf

```
void explain_message_errno_setbuf(char *message, int message_size, int errnum, FILE *fp, char *data);
```

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

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*.

fp

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

data

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

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

```

errno = 0;
void result = setbuf(fp, data);
if (result < 0 && errno != 0)
{

```



explain_setbuf(3)

explain_setbuf(3)

```
    int err = errno;
    char message[3000];
    explain_message_errno_setbuf(message, sizeof(message), err,
    fp, data);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

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

SEE ALSO*setbuf(3)*

set stream buffer

explain_setbuf_or_die(3)

set stream buffer and report errors

COPYRIGHT

libexplain version 1.4

Copyright © 2010 Peter Miller

