

explain_flock(3)

explain_flock(3)

NAME

explain_flock – explain flock(2) errors

SYNOPSIS

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

const char *explain_flock(int fildes, int command);
const char *explain_errno_flock(int errnum, int fildes, int command);
void explain_message_flock(char *message, int message_size, int fildes, int command);
void explain_message_errno_flock(char *message, int message_size, int errnum, int fildes, int command);
```

DESCRIPTION

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

explain_flock

```
const char *explain_flock(int fildes, int command);
```

The **explain_flock** function is used to obtain an explanation of an error returned by the *flock(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 *flock(2)* system call.

command

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

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

explain_errno_flock

```
const char *explain_errno_flock(int errnum, int fildes, int command);
```

The **explain_errno_flock** function is used to obtain an explanation of an error returned by the *flock(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 *flock(2)* system call.

command

The original command, exactly as passed to the *flock(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 (flock(fildes, command) < 0)
{
```



explain_flock(3)

explain_flock(3)

```

        int err = errno;
        fprintf(stderr, "%s\n", explain_errno_flock(err, fildes,
        command));
        exit(EXIT_FAILURE);
    }

```

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

explain_message_flock

```
void explain_message_flock(char *message, int message_size, int fildes, int command);
```

The **explain_message_flock** function is used to obtain an explanation of an error returned by the *flock*(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 *flock*(2) system call.

command

The original command, exactly as passed to the *flock*(2) system call.

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

```

if (flock(fildes, command) < 0)
{
    char message[3000];
    explain_message_flock(message, sizeof(message), fildes,
    command);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}

```

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

explain_message_errno_flock

```
void explain_message_errno_flock(char *message, int message_size, int errnum, int fildes, int command);
```

The **explain_message_errno_flock** function is used to obtain an explanation of an error returned by the *flock*(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 *flock*(2) system call.

command

The original command, exactly as passed to the *flock*(2) system call.

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

```

if (flock(fildes, command) < 0)
{
    int err = errno;
    char message[3000];
    explain_message_errno_flock(message, sizeof(message), err,

```



explain_flock(3)

explain_flock(3)

```
        fildes, command);  
        fprintf(stderr, "%s\n", message);  
        exit(EXIT_FAILURE);  
    }
```

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

SEE ALSO

flock(2) apply or remove an advisory lock on an open file

explain_flock_or_die(3)

apply or remove an advisory lock on an open file and report errors

COPYRIGHT

libexplain version 1.4

Copyright © 2009 Peter Miller

