NAME

explain_utimensat − explain *utimensat*(2) errors

SYNOPSIS

#include <libexplain/utimensat.h>

const char *explain_utimensat(int fildes, const char *pathname, const struct timespec *data, int flags);
const char *explain_errno_utimensat(int errnum, int fildes, const char *pathname, const struct timespec *data, int flags);
void explain_message_utimensat(char *message, int message_size, int fildes, const char *pathname, const struct timespec *data, int flags);
void explain_message_errno_utimensat(char *message, int message_size, int errnum, int fildes, const char *pathname, const struct timespec *data, int flags);

DESCRIPTION

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

explain_utimensat

const char *explain_utimensat(int fildes, const char *pathname, const struct timespec *data, int flags);

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

pathname
The original pathname, exactly as passed to the *utimensat*(2) system call.

data The original data, exactly as passed to the *utimensat*(2) system call.

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

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

explain_errno_utimensat

const char *explain_errno_utimensat(int errnum, int fildes, const char *pathname, const struct timespec *data, int flags);

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

pathname
The original pathname, exactly as passed to the *utimensat*(2) system call.

data The original data, exactly as passed to the *utimensat*(2) system call.

flags The original flags, exactly as passed to the *utimensat*(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 (utimensat(fildes, pathname, data, flags) < 0)
{
     int err = errno;
    fprintf(stderr, "%s\n", explain_errno_utimensat(err,
    fildes, pathname, data, flags));
     exit(EXIT_FAILURE);
}
```

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

explain_message_utimensat

void explain_message_utimensat(char *message, int message_size, int fildes, const char *pathname, const struct timespec *data, int flags);

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

pathname
The original pathname, exactly as passed to the *utimensat*(2) system call.

data The original data, exactly as passed to the *utimensat*(2) system call.

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

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

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

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

explain_message_errno_utimensat

void explain_message_errno_utimensat(char *message, int message_size, int errnum, int fildes, const char *pathname, const struct timespec *data, int flags);

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

pathname
The original pathname, exactly as passed to the *utimensat*(2) system call.

data The original data, exactly as passed to the *utimensat*(2) system call.

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

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

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

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

SEE ALSO

utimensat(2)
change file timestamps with nanosecond precision

explain_utimensat_or_die(3)
change file timestamps with nanosecond precision and report errors

COPYRIGHT