NAME

explain_mknod − explain *mknod*(2) errors

SYNOPSIS

#include <libexplain/mknod.h>

const char *explain_mknod(const char *pathname, mode_t mode, dev_t dev);
const char *explain_errno_mknod(int errnum, const char *pathname, mode_t mode, dev_t dev);
void explain_message_mknod(char *message, int message_size, const char *pathname, mode_t mode, dev_t dev);
void explain_message_errno_mknod(char *message, int message_size, int errnum, const char *pathname, mode_t mode, dev_t dev);

DESCRIPTION

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

explain_mknod

const char *explain_mknod(const char *pathname, mode_t mode, dev_t dev);

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

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

mode
: The original mode, exactly as passed to the *mknod*(2) system call.

dev
: The original dev, exactly as passed to the *mknod*(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 (mknod(pathname, mode, dev) < 0)
{
    fprintf(stderr, "%s\n", explain_mknod(pathname, mode,
    dev));
     exit(EXIT_FAILURE);
}
```

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

explain_errno_mknod

const char *explain_errno_mknod(int errnum, const char *pathname, mode_t mode, dev_t dev);

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

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

mode
: The original mode, exactly as passed to the *mknod*(2) system call.

dev
: The original dev, exactly as passed to the *mknod*(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 (mknod(pathname, mode, dev) < 0)
{
    int err = errno;
   fprintf(stderr, "%s\n", explain_errno_mknod(err, pathname,
   mode, dev));
    exit(EXIT_FAILURE);
}
```

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

explain_message_mknod

void explain_message_mknod(char *message, int message_size, const char *pathname, mode_t mode, dev_t dev);

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

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

mode The original mode, exactly as passed to the *mknod*(2) system call.

dev The original dev, exactly as passed to the *mknod*(2) system call.

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

```
if (mknod(pathname, mode, dev) < 0)
{
    char message[3000];
   explain_message_mknod(message, sizeof(message), pathname,
   mode, dev);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

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

explain_message_errno_mknod

void explain_message_errno_mknod(char *message, int message_size, int errnum, const char *pathname, mode_t mode, dev_t dev);

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

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

mode The original mode, exactly as passed to the *mknod*(2) system call.

dev The original dev, exactly as passed to the *mknod*(2) system call.

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

```
if (mknod(pathname, mode, dev) < 0)
{
    int err = errno;
    char message[3000];
   explain_message_errno_mknod(message, sizeof(message), err,
   pathname, mode, dev);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

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

SEE ALSO

mknod(2)
create a special or ordinary file

explain_mknod_or_die(3)
create a special or ordinary file and report errors

COPYRIGHT