

explain\_endgrent(3)

explain\_endgrent(3)

**NAME**explain\_endgrent – explain *endgrent*(3) errors**SYNOPSIS**

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

const char *explain_endgrent(void);
const char *explain_errno_endgrent(int errnum, void);
void explain_message_endgrent(char *message, int message_size, void);
void explain_message_errno_endgrent(char *message, int message_size, int errnum, void);
```

**DESCRIPTION**

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

**explain\_endgrent**

```
const char *explain_endgrent(void);
```

The **explain\_endgrent** function is used to obtain an explanation of an error returned by the *endgrent*(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.

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 = endgrent();
if (result < 0 && errno != 0)
{
    fprintf(stderr, "%s\n", explain_endgrent());
    exit(EXIT_FAILURE);
}
```

The above code example is available pre-packaged as the *explain\_endgrent\_or\_die*(3) function.

**explain\_errno\_endgrent**

```
const char *explain_errno_endgrent(int errnum, void);
```

The **explain\_errno\_endgrent** function is used to obtain an explanation of an error returned by the *endgrent*(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*.

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



explain\_endgrent(3)

explain\_endgrent(3)

The above code example is available pre-packaged as the *explain\_endgrent\_or\_die(3)* function.

### explain\_message\_endgrent

```
void explain_message_endgrent(char *message, int message_size, void);
```

The **explain\_message\_endgrent** function is used to obtain an explanation of an error returned by the *endgrent(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.

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

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

The above code example is available pre-packaged as the *explain\_endgrent\_or\_die(3)* function.

### explain\_message\_errno\_endgrent

```
void explain_message_errno_endgrent(char *message, int message_size, int errnum, void);
```

The **explain\_message\_errno\_endgrent** function is used to obtain an explanation of an error returned by the *endgrent(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*.

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

```
errno = 0;
void result = endgrent();
if (result < 0 && errno != 0)
{
    int err = errno;
    char message[3000];
    explain_message_errno_endgrent(message, sizeof(message),
    err, );
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

The above code example is available pre-packaged as the *explain\_endgrent\_or\_die(3)* function.

## SEE ALSO

*endgrent(3)*

finish group file accesses



explain\_endgrent(3)

explain\_endgrent(3)

*explain\_endgrent\_or\_die(3)*

finish group file accesses and report errors

## **COPYRIGHT**

libexplain version 1.4

Copyright © 2013 Peter Miller

