NAME

explain_time − explain time(2) errors

SYNOPSIS

#include <libexplain/time.h>

const char *explain_time(time_t *t);
const char *explain_errno_time(int errnum, time_t *t);
void explain_message_time(char *message, int message_size, time_t *t);
void explain_message_errno_time(char *message, int message_size, int errnum, time_t *t);

DESCRIPTION

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

explain_time

const char *explain_time(time_t *t);

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

t The original t, exactly as passed to the *time*(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:

```
time_t result = time(t);
if (result == (time_t)-1)
{
    fprintf(stderr, "%s\n", explain_time(t));
     exit(EXIT_FAILURE);
}
```

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

explain_errno_time

const char *explain_errno_time(int errnum, time_t *t);

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

t The original t, exactly as passed to the *time*(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:

```
time_t result = time(t);
if (result == (time_t)-1)
{
     int err = errno;
    fprintf(stderr, "%s\n", explain_errno_time(err, t));
     exit(EXIT_FAILURE);
}
```

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

explain_message_time

void explain_message_time(char *message, int message_size, time_t *t);

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

t The original t, exactly as passed to the *time*(2) system call.

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

```
time_t result = time(t);
if (result == (time_t)-1)
{
    char message[3000];
   explain_message_time(message, sizeof(message), t);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

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

explain_message_errno_time

void explain_message_errno_time(char *message, int message_size, int errnum, time_t *t);

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

t The original t, exactly as passed to the *time*(2) system call.

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

```
time_t result = time(t);
if (result == (time_t)-1)
{
    int err = errno;
    char message[3000];
   explain_message_errno_time(message, sizeof(message), err,
   t);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

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

SEE ALSO

time(2) get time in seconds

explain_time_or_die(3)
get time in seconds and report errors

COPYRIGHT