## NAME

explain_putw − explain *putw*(3) errors

## SYNOPSIS

#include <libexplain/putw.h>

const char *explain_putw(int value, FILE *fp);
const char *explain_errno_putw(int errnum, int value, FILE *fp);
void explain_message_putw(char *message, int message_size, int value, FILE *fp);
void explain_message_errno_putw(char *message, int message_size, int errnum, int value, FILE *fp);

## DESCRIPTION

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

### explain_putw

const char *explain_putw(int value, FILE *fp);

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

*value* The original value, exactly as passed to the *putw*(3) system call.

*fp* The original fp, exactly as passed to the *putw*(3) 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 (putw(value, fp) < 0)
{
    fprintf(stderr, "%s\n", explain_putw(value, fp));
     exit(EXIT_FAILURE);
}
```

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

### explain_errno_putw

const char *explain_errno_putw(int errnum, int value, FILE *fp);

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

*value* The original value, exactly as passed to the *putw*(3) system call.

*fp* The original fp, exactly as passed to the *putw*(3) 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 (putw(value, fp) < 0)
{
     int err = errno;
    fprintf(stderr, "%s\n", explain_errno_putw(err, value,
    fp));
```

```
        exit(EXIT_FAILURE);
}
```

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

**explain_message_putw**

void explain_message_putw(char *message, int message_size, int value, FILE *fp);

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

*value* The original value, exactly as passed to the *putw*(3) system call.

*fp* The original fp, exactly as passed to the *putw*(3) system call.

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

```
if (putw(value, fp) < 0)
{
    char message[3000];
   explain_message_putw(message, sizeof(message), value, fp);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

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

**explain_message_errno_putw**

void explain_message_errno_putw(char *message, int message_size, int errnum, int value, FILE *fp);

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

*value* The original value, exactly as passed to the *putw*(3) system call.

*fp* The original fp, exactly as passed to the *putw*(3) system call.

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

```
if (putw(value, fp) < 0)
{
    int err = errno;
    char message[3000];
   explain_message_errno_putw(message, sizeof(message), err,
   value, fp);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

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

## SEE ALSO

*putw*(3) output a word (int)

*explain_putw_or_die*(3)
output a word (int) and report errors

## COPYRIGHT