

explain\_strcoll(3)

explain\_strcoll(3)

**NAME**explain\_strcoll – explain *strcoll*(3) errors**SYNOPSIS**

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

const char *explain_strcoll(const char *s1, const char *s2);
const char *explain_errno_strcoll(int errnum, const char *s1, const char *s2);
void explain_message_strcoll(char *message, int message_size, const char *s1, const char *s2);
void explain_message_errno_strcoll(char *message, int message_size, int errnum, const char *s1, const char *s2);
```

**DESCRIPTION**

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

**explain\_strcoll**

```
const char *explain_strcoll(const char *s1, const char *s2);
```

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

*s1*        The original *s1*, exactly as passed to the *strcoll*(3) system call.

*s2*        The original *s2*, exactly as passed to the *strcoll*(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:

```
errno = 0;
int result = strcoll(s1, s2);
if (result < 0 && errno != 0)
{
    fprintf(stderr, "%s\n", explain_strcoll(s1, s2));
    exit(EXIT_FAILURE);
}
```

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

**explain\_errno\_strcoll**

```
const char *explain_errno_strcoll(int errnum, const char *s1, const char *s2);
```

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

*s1*        The original *s1*, exactly as passed to the *strcoll*(3) system call.

*s2*        The original *s2*, exactly as passed to the *strcoll*(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:

```
errno = 0;
int result = strcoll(s1, s2);
```



explain\_strcoll(3)

explain\_strcoll(3)

```

if (result < 0 && errno != 0)
{
    int err = errno;
    fprintf(stderr, "%s\n", explain_errno_strcoll(err, s1,
s2));
    exit(EXIT_FAILURE);
}

```

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

### explain\_message\_strcoll

```
void explain_message_strcoll(char *message, int message_size, const char *s1, const char *s2);
```

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

*s1* The original *s1*, exactly as passed to the *strcoll(3)* system call.

*s2* The original *s2*, exactly as passed to the *strcoll(3)* system call.

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

```

errno = 0;
int result = strcoll(s1, s2);
if (result < 0 && errno != 0)
{
    char message[3000];
    explain_message_strcoll(message, sizeof(message), s1, s2);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}

```

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

### explain\_message\_errno\_strcoll

```
void explain_message_errno_strcoll(char *message, int message_size, int errnum, const char *s1, const char *s2);
```

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

*s1* The original *s1*, exactly as passed to the *strcoll(3)* system call.

*s2* The original *s2*, exactly as passed to the *strcoll(3)* system call.

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

```

errno = 0;
int result = strcoll(s1, s2);
if (result < 0 && errno != 0)
{

```



explain\_strcoll(3)

explain\_strcoll(3)

```
    int err = errno;
    char message[3000];
    explain_message_errno_strcoll(message, sizeof(message),
    err, s1, s2);
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}
```

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

**SEE ALSO***strcoll*(3)

compare two strings using the current locale

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

compare two strings using the current locale and report errors

**COPYRIGHT**

libexplain version 1.4

Copyright © 2013 Peter Miller

