Rocky Enterprise Linux 9.2 Manual Pages on command 'freelocale.3'

$ man freelocale.3

NEWLOCALE(3) Linux Programmer's Manual NEWLOCALE(3)

NAME

newlocale, freelocale - create, modify, and free a locale object

SYNOPSIS

#include <locale.h>

locale_t newlocale(int category_mask, const char *locale,
locale_t base);

void freelocale(locale_t locobj);

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

newlocale(), freelocale():

Since glibc 2.10:

_XOPEN_SOURCE >= 700

Before glibc 2.10:

_GNU_SOURCE

DESCRIPTION

The newlocale() function creates a new locale object, or modifies an existing object, returning a reference to the new or modified object as the function result. Whether the call creates a new object or modifies

an existing object is determined by the value of base:

* If base is (locale_t) 0, a new object is created.

* If base refers to valid existing locale object (i.e., an object re?
  turned by a previous call to newlocale() or duplocale(3)), then that
  object is modified by the call. If the call is successful, the con?
  tents of base are unspecified (in particular, the object referred to
  by base may be freed, and a new object created). Therefore, the
  caller should ensure that it stops using base before the call to
  newlocale(), and should subsequently refer to the modified object
  via the reference returned as the function result. If the call
  fails, the contents of base remain valid and unchanged.

If base is the special locale object LC_GLOBAL_LOCALE (see duplo?
cale(3)), or is not (locale_t) 0 and is not a valid locale object han?
dle, the behavior is undefined.

The category_mask argument is a bit mask that specifies the locale cat?
egories that are to be set in a newly created locale object or modified
in an existing object. The mask is constructed by a bitwise OR of the
constants LC_ADDRESS_MASK, LC_CTYPE_MASK, LC_COLLATE_MASK, LC_IDENTIFI?
CATION_MASK, LC_MEASUREMENT_MASK, LC_MESSAGES_MASK, LC_MONETARY_MASK,
LC_NUMERIC_MASK, LC_NAME_MASK, LC_PAPER_MASK, LC_TELEPHONE_MASK, and
LC_TIME_MASK. Alternatively, the mask can be specified as LC_ALL_MASK,
which is equivalent to ORing all of the preceding constants.

For each category specified in category_mask, the locale data from lo?
cale will be used in the object returned by newlocale(). If a new lo?
cale object is being created, data for all categories not specified in
category_mask is taken from the default ("POSIX") locale.

The following preset values of locale are defined for all categories
that can be specified in category_mask:

"POSIX"
    A minimal locale environment for C language programs.

"C" Equivalent to "POSIX".

"" An implementation-defined native environment corresponding to
    the values of the LC_* and LANG environment variables (see lo?

cale(7)).

freelocale()

The freelocale() function deallocates the resources associated with lo?
cobj, a locale object previously returned by a call to newlocale() or
duplocale(3). If locobj is LC_GLOBAL_LOCALE or is not valid locale ob?
ject handle, the results are undefined.

Once a locale object has been freed, the program should make no further
use of it.

RETURN VALUE

On success, newlocale() returns a handle that can be used in calls to
duplocale(3), freelocale(), and other functions that take a locale_t
argument. On error, newlocale() returns (locale_t) 0, and sets errno
to indicate the cause of the error.

ERRORS

EINVAL One or more bits in category_mask do not correspond to a valid
locale category.

EINVAL locale is NULL.

ENOENT locale is not a string pointer referring to a valid locale.

ENOMEM Insufficient memory to create a locale object.

VERSIONS

The newlocale() and freelocale() functions first appeared in version
2.3 of the GNU C library.

CONFORMING TO

POSIX.1-2008.

NOTES

Each locale object created by newlocale() should be deallocated using
freelocale().

EXAMPLES

The program below takes up to two command-line arguments, which each
identify locales. The first argument is required, and is used to set
the LC_NUMERIC category in a locale object created using newlocale().
The second command-line argument is optional; if it is present, it is
used to set the LC_TIME category of the locale object.

Having created and initialized the locale object, the program then ap?
plies it using uselocale(3), and then tests the effect of the locale
changes by:

1. Displaying a floating-point number with a fractional part. This
output will be affected by the LC_NUMERIC setting. In many Euro?
pean-language locales, the fractional part of the number is sepa?
rated from the integer part using a comma, rather than a period.
2. Displaying the date. The format and language of the output will be
affected by the LC_TIME setting.

The following shell sessions show some example runs of this program.

Set the LC_NUMERIC category to fr_FR (French):

```
$ ./a.out fr_FR
123456,789
Fri Mar  7 00:25:08 2014
```

Set the LC_NUMERIC category to fr_FR (French), and the LC_TIME category
to it_IT (Italian):

```
$ ./a.out fr_FR it_IT
123456,789
ven 07 mar 2014 00:26:01 CET
```

Specify the LC_TIME setting as an empty string, which causes the value
to be taken from environment variable settings (which, here, specify
mi_NZ, New Zealand M?ori):

```
$ LC_ALL=mi_NZ ./a.out fr_FR ""
123456,789
Te Paraire, te 07 o Pout?-te-rangi, 2014 00:38:44 CET
```

Program source

```
#define _XOPEN_SOURCE 700
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <time.h>
#define errExit(msg)    do { perror(msg); exit(EXIT_FAILURE); \
                        } while (0)
```

```
int
main(int argc, char *argv[])
{
    char buf[100];
    time_t t;
    size_t s;
    struct tm *tm;
    locale_t loc, nloc;
    if (argc < 2) {
        fprintf(stderr, "Usage: %s locale1 [locale2]\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    /* Create a new locale object, taking the LC_NUMERIC settings
       from the locale specified in argv[1] */
    loc = newlocale(LC_NUMERIC_MASK, argv[1], (locale_t) 0);
    if (loc == (locale_t) 0)
        errExit("newlocale");
    /* If a second command-line argument was specified, modify the
       locale object to take the LC_TIME settings from the locale
       specified in argv[2]. We assign the result of this newlocale()
       call to 'nloc' rather than 'loc', since in some cases, we might
       want to preserve 'loc' if this call fails. */
    if (argc > 2) {
        nloc = newlocale(LC_TIME_MASK, argv[2], loc);
        if (nloc == (locale_t) 0)
            errExit("newlocale");
        loc = nloc;
    }
    /* Apply the newly created locale to this thread */
    uselocale(loc);
    /* Test effect of LC_NUMERIC */
    printf("%8.3f\n", 123456.789);
    /* Test effect of LC_TIME */
```

```
        t = time(NULL);
        tm = localtime(&t);
        if (tm == NULL)
            errExit("time");
        s = strftime(buf, sizeof(buf), "%c", tm);
        if (s == 0)
            errExit("strftime");
        printf("%s\n", buf);
        /* Free the locale object */
        uselocale(LC_GLOBAL_HANDLE);    /* So 'loc' is no longer in use */
        freelocale(loc);
        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

locale(1), duplocale(3), setlocale(3), uselocale(3), locale(5), lo?
cale(7)

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.