Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'setlocale.3p' command

$ man setlocale.3p

SETLOCALE(3P) POSIX Programmer's Manual SETLOCALE(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

setlocale ? set program locale

SYNOPSIS

#include <locale.h>

char *setlocale(int category, const char *locale);

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1?2017 defers to the ISO C standard.

The setlocale() function selects the appropriate piece of the global locale, as specified by the category and locale arguments, and can be used to change or query the entire global locale or portions thereof.

The value LC_ALL for category names the entire global locale; other values for category name only a part of the global locale:

LC_COLLATE Affects the behavior of regular expressions and the colla?

tion functions.

LC_CTYPE Affects the behavior of regular expressions, character classification, character conversion functions, and wide-character functions.

LC_MESSAGES Affects the affirmative and negative response expressions returned by nl_langinfo() and the way message catalogs are located. It may also affect the behavior of functions that return or write message strings.

LC_MONETARY Affects the behavior of functions that handle monetary val? ues.

LC_NUMERIC Affects the behavior of functions that handle numeric val? ues.

LC_TIME Affects the behavior of the time conversion functions.

The locale argument is a pointer to a character string containing the required setting of category. The contents of this string are imple? mentation-defined. In addition, the following preset values of locale are defined for all settings of category:

"POSIX" Specifies the minimal environment for C-language transla? tion called the POSIX locale. The POSIX locale is the de? fault global locale at entry to main().

"C" Equivalent to "POSIX".

"" Specifies an implementation-defined native environment. The determination of the name of the new locale for the specified category depends on the value of the associated environment variables, LC_* and LANG; see the Base Defini? tions volume of POSIX.1?2017, Chapter 7, Locale and Chapter 8, Environment Variables.

A null pointer

Directs setlocale() to query the current global locale set? ting and return the name of the locale if category is not LC_ALL, or a string which encodes the locale name(s) for all of the individual categories if category is LC_ALL.

Setting all of the categories of the global locale is similar to suc? cessively setting each individual category of the global locale, except

that all error checking is done before any actions are performed. To set all the categories of the global locale, setlocale() can be invoked as:

```
setlocale(LC_ALL, "");
```

In this case, setlocale() shall first verify that the values of all the environment variables it needs according to the precedence rules (described in the Base Definitions volume of POSIX.1?2017, Chapter 8, Environment Variables) indicate supported locales. If the value of any of these environment variable searches yields a locale that is not supported (and non-null), setlocale() shall return a null pointer and the global locale shall not be changed. If all environment variables name supported locales, setlocale() shall proceed as if it had been called for each category, using the appropriate value from the associated environment variable or from the implementation-defined default if there is no such value.

The global locale established using setlocale() shall only be used in threads for which no current locale has been set using uselocale() or whose current locale has been set to the global locale using uselocale(LC_GLOBAL_LOCALE).

The implementation shall behave as if no function defined in this volume of POSIX.1?2017 calls setlocale().

The setlocale() function need not be thread-safe.

RETURN VALUE

Upon successful completion, setlocale() shall return the string associated with the specified category for the new locale. Otherwise, setlocale() shall return a null pointer and the global locale shall not be changed.

A null pointer for locale shall cause setlocale() to return a pointer to the string associated with the specified category for the current global locale. The global locale shall not be changed.

The string returned by setlocale() is such that a subsequent call with that string and its associated category shall restore that part of the global locale. The application shall not modify the string returned.

The returned string pointer might be invalidated or the string content might be overwritten by a subsequent call to setlocale(). The returned pointer might also be invalidated if the calling thread is terminated.

ERRORS

No errors are defined.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

The following code illustrates how a program can initialize the inter? national environment for one language, while selectively modifying the global locale such that regular expressions and string operations can be applied to text recorded in a different language:

```
setlocale(LC_ALL, "De");
setlocale(LC_COLLATE, "Fr@dict");
```

Internationalized programs can initiate language operation according to environment variable settings (see the Base Definitions volume of POSIX.1?2017, Section 8.2, Internationalization Variables) by calling setlocale() as follows:

```
setlocale(LC_ALL, "");
```

Changing the setting of LC_MESSAGES has no effect on catalogs that have already been opened by calls to catopen().

In order to make use of different locale settings while multiple threads are running, applications should use uselocale() in preference to setlocale().

RATIONALE

References to the international environment or locale in the following text relate to the global locale for the process. This can be overrid? den for individual threads using uselocale().

The ISO C standard defines a collection of functions to support inter? nationalization. One of the most significant aspects of these func? tions is a facility to set and query the international environment.

The international environment is a repository of information that af?

fects the behavior of certain functionality, namely:

1. Character handling
2. Collating
3. Date/time formatting
4. Numeric editing
5. Monetary formatting
6. Messaging

The setlocale() function provides the application developer with the ability to set all or portions, called categories, of the international environment. These categories correspond to the areas of functionality mentioned above. The syntax for setlocale() is as follows:

```
char *setlocale(int category, const char *locale);
```

where category is the name of one of following categories, namely:

```
LC_COLLATE LC_CTYPE LC_MESSAGES LC_MONETARY LC_NUMERIC LC_TIME
```

In addition, a special value called LC_ALL directs setlocale() to set all categories.

There are two primary uses of setlocale():

1. Querying the international environment to find out what it is set to
2. Setting the international environment, or locale, to a specific value

The behavior of setlocale() in these two areas is described below. Since it is difficult to describe the behavior in words, examples are used to illustrate the behavior of specific uses.

To query the international environment, setlocale() is invoked with a specific category and the null pointer as the locale. The null pointer is a special directive to setlocale() that tells it to query rather than set the international environment. The following syntax is used to query the name of the international environment:

```
setlocale({LC_ALL, LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, \
    LC_NUMERIC, LC_TIME},(char *) NULL);
```

The setlocale() function shall return the string corresponding to the current international environment. This value may be used by a subse?

quent call to setlocale() to reset the international environment to this value. However, it should be noted that the return value from set? locale() may be a pointer to a static area within the function and is not guaranteed to remain unchanged (that is, it may be modified by a subsequent call to setlocale()). Therefore, if the purpose of calling setlocale() is to save the value of the current international environ? ment so it can be changed and reset later, the return value should be copied to an array of char in the calling program.

There are three ways to set the international environment with setlo? cale():

setlocale(category, string)

This usage sets a specific category in the international environ? ment to a specific value corresponding to the value of the string. A specific example is provided below:

```
setlocale(LC_ALL, "fr_FR.ISO-8859-1");
```

In this example, all categories of the international environment are set to the locale corresponding to the string "fr_FR.ISO-8859-1", or to the French language as spoken in France using the ISO/IEC 8859?1:1998 standard codeset.

If the string does not correspond to a valid locale, setlocale() shall return a null pointer and the international environment is not changed. Otherwise, setlocale() shall return the name of the locale just set.

setlocale(category, "C")

The ISO C standard states that one locale must exist on all con? forming implementations. The name of the locale is C and corre? sponds to a minimal international environment needed to support the C programming language.

setlocale(category, "")

This sets a specific category to an implementation-defined de? fault. This corresponds to the value of the environment vari? ables.

None.

SEE ALSO

catopen(), exec, fprintf(), fscanf(), isalnum(), isalpha(), isblank(), iscntrl(), isdigit(), isgraph(), islower(), isprint(), ispunct(), iss? pace(), isupper(), iswalnum(), iswalpha(), iswblank(), iswcntrl(), iswctype(), iswdigit(), iswgraph(), iswlower(), iswprint(), iswpunct(), iswspace(), iswupper(), iswxdigit(), isxdigit(), localeconv(), mblen(), mbstowcs(), mbtowc(), newlocale(), nl_langinfo(), perror(), psiginfo(), strcoll(), strerror(), strfmon(), strsignal(), strtod(), strxfrm(), tolower(), toupper(), towlower(), towupper(), uselocale(), wcscoll(), wcstod(), wcstombs(), wcsxfrm(), wctomb()

The Base Definitions volume of POSIX.1?2017, Chapter 7, Locale, Chapter 8, Environment Variables, <langinfo.h>, <locale.h>

COPYRIGHT