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 'locale.7' command

$ man locale.7

LOCALE(7) Linux Programmer's Manual LOCALE(7)

NAME

locale - description of multilanguage support

SYNOPSIS

#include <locale.h>

DESCRIPTION

A locale is a set of language and cultural rules. These cover aspects such as language for messages, different character sets, lexicographic conventions, and so on. A program needs to be able to determine its locale and act accordingly to be portable to different cultures.

The header <locale.h> declares data types, functions and macros which are useful in this task.

The functions it declares are setlocale(3) to set the current locale, and localeconv(3) to get information about number formatting.

There are different categories for locale information a program might need; they are declared as macros. Using them as the first argument to the setlocale(3) function, it is possible to set one of these to the desired locale:

LC_ADDRESS (GNU extension, since glibc 2.2)

Change settings that describe the formats (e.g., postal ad? dresses) used to describe locations and geography-related items. Applications that need this information can use nl_langinfo(3) to retrieve nonstandard elements, such as _NL_ADDRESS_COUN?

TRY_NAME (country name, in the language of the locale) and _NL_ADDRESS_LANG_NAME (language name, in the language of the lo? cale), which return strings such as "Deutschland" and "Deutsch" (for German-language locales). (Other element names are listed in <langinfo.h>.)

LC_COLLATE

This category governs the collation rules used for sorting and regular expressions, including character equivalence classes and multicharacter collating elements. This locale category changes the behavior of the functions strcoll(3) and strxfrm(3), which are used to compare strings in the local alphabet. For example, the German sharp s is sorted as "ss".

LC_CTYPE

This category determines the interpretation of byte sequences as characters (e.g., single versus multibyte characters), character classifications (e.g., alphabetic or digit), and the behavior of character classes. On glibc systems, this category also deter? mines the character transliteration rules for iconv(1) and iconv(3). It changes the behavior of the character handling and classification functions, such as isupper(3) and toupper(3), and the multibyte character functions such as mblen(3) or wctomb(3).

LC_IDENTIFICATION (GNU extension, since glibc 2.2)

Change settings that relate to the metadata for the locale. Ap? plications that need this information can use nl_langinfo(3) to retrieve nonstandard elements, such as _NL_IDENTIFICATION_TITLE (title of this locale document) and _NL_IDENTIFICATION_TERRITORY (geographical territory to which this locale document applies), which might return strings such as "English locale for the USA" and "USA". (Other element names are listed in <langinfo.h>.)

LC_MONETARY

This category determines the formatting used for monetary-re? lated numeric values. This changes the information returned by localeconv(3), which describes the way numbers are usually

printed, with details such as decimal point versus decimal comma. This information is internally used by the function strfmon(3).

LC_MESSAGES

This category affects the language in which messages are dis? played and what an affirmative or negative answer looks like. The GNU C library contains the gettext(3), ngettext(3), and rp? match(3) functions to ease the use of this information. The GNU gettext family of functions also obey the environment variable LANGUAGE (containing a colon-separated list of locales) if the category is set to a valid locale other than "C". This category also affects the behavior of catopen(3).

LC_MEASUREMENT (GNU extension, since glibc 2.2)

Change the settings relating to the measurement system in the locale (i.e., metric versus US customary units). Applications can use nl_langinfo(3) to retrieve the nonstandard _NL_MEASURE? MENT_MEASUREMENT element, which returns a pointer to a character that has the value 1 (metric) or 2 (US customary units).

LC_NAME (GNU extension, since glibc 2.2)

Change settings that describe the formats used to address per? sons. Applications that need this information can use nl_lang? info(3) to retrieve nonstandard elements, such as _NL_NAME_NAME_MR (general salutation for men) and _NL_NAME_NAME_MS (general salutation for women) elements, which return strings such as "Herr" and "Frau" (for German-language locales). (Other element names are listed in <langinfo.h>.)

LC_NUMERIC

This category determines the formatting rules used for nonmone? tary numeric values?for example, the thousands separator and the radix character (a period in most English-speaking countries, but a comma in many other regions). It affects functions such as printf(3), scanf(3), and strtod(3). This information can also be read with the localeconv(3) function.

LC_PAPER (GNU extension, since glibc 2.2)

Change the settings relating to the dimensions of the standard paper size (e.g., US letter versus A4). Applications that need the dimensions can obtain them by using nl_langinfo(3) to re? trieve the nonstandard _NL_PAPER_WIDTH and _NL_PAPER_HEIGHT ele? ments, which return int values specifying the dimensions in mil? limeters.

LC_TELEPHONE (GNU extension, since glibc 2.2)

Change settings that describe the formats to be used with tele? phone services. Applications that need this information can use nl_langinfo(3) to retrieve nonstandard elements, such as _NL_TELEPHONE_INT_PREFIX (international prefix used to call num? bers in this locale), which returns a string such as "49" (for Germany). (Other element names are listed in <langinfo.h>.)

LC_TIME

This category governs the formatting used for date and time val? ues. For example, most of Europe uses a 24-hour clock versus the 12-hour clock used in the United States. The setting of this category affects the behavior of functions such as strf? time(3) and strptime(3).

LC_ALL All of the above.

If the second argument to setlocale(3) is an empty string, "", for the default locale, it is determined using the following steps:

1. If there is a non-null environment variable LC_ALL, the value of LC_ALL is used.
2. If an environment variable with the same name as one of the cate? gories above exists and is non-null, its value is used for that cat? egory.
3. If there is a non-null environment variable LANG, the value of LANG is used.

Values about local numeric formatting is made available in a struct lconv returned by the localeconv(3) function, which has the following declaration:

```
struct lconv {
    /* Numeric (nonmonetary) information */
    char *decimal_point;     /* Radix character */
    char *thousands_sep;     /* Separator for digit groups to left
                                of radix character */
    char *grouping;     /* Each element is the number of digits in
                           a group; elements with higher indices
                           are further left.  An element with value
                           CHAR_MAX means that no further grouping
                           is done.  An element with value 0 means
                           that the previous element is used for
                           all groups further left. */
    /* Remaining fields are for monetary information */
    char *int_curr_symbol;   /* First three chars are a currency
                                symbol from ISO 4217.  Fourth char
                                is the separator.  Fifth char
                                is '\0'. */
    char *currency_symbol;   /* Local currency symbol */
    char *mon_decimal_point; /* Radix character */
    char *mon_thousands_sep; /* Like thousands_sep above */
    char *mon_grouping;      /* Like grouping above */
    char *positive_sign;     /* Sign for positive values */
    char *negative_sign;     /* Sign for negative values */
    char  int_frac_digits;   /* International fractional digits */
    char  frac_digits;       /* Local fractional digits */
    char  p_cs_precedes;     /* 1 if currency_symbol precedes a
                                positive value, 0 if succeeds */
    char  p_sep_by_space;    /* 1 if a space separates
                                currency_symbol from a positive
                                value */
    char  n_cs_precedes;     /* 1 if currency_symbol precedes a
                                negative value, 0 if succeeds */
    char  n_sep_by_space;    /* 1 if a space separates
```

```
                    currency_symbol from a negative
                    value */
        /* Positive and negative sign positions:
           0 Parentheses surround the quantity and currency_symbol.
           1 The sign string precedes the quantity and currency_symbol.
           2 The sign string succeeds the quantity and currency_symbol.
           3 The sign string immediately precedes the currency_symbol.
           4 The sign string immediately succeeds the currency_symbol. */
        char  p_sign_posn;
        char  n_sign_posn;
    };
```

POSIX.1-2008 extensions to the locale API

POSIX.1-2008 standardized a number of extensions to the locale API, based on implementations that first appeared in version 2.3 of the GNU C library. These extensions are designed to address the problem that the traditional locale APIs do not mix well with multithreaded applica? tions and with applications that must deal with multiple locales.

The extensions take the form of new functions for creating and manipu? lating locale objects (newlocale(3), freelocale(3), duplocale(3), and uselocale(3)) and various new library functions with the suffix "_l" (e.g., toupper_l(3)) that extend the traditional locale-dependent APIs (e.g., toupper(3)) to allow the specification of a locale object that should apply when executing the function.

ENVIRONMENT

The following environment variable is used by newlocale(3) and setlo? cale(3), and thus affects all unprivileged localized programs:

LOCPATH

A list of pathnames, separated by colons (':'), that should be used to find locale data. If this variable is set, only the in? dividual compiled locale data files from LOCPATH and the system default locale data path are used; any available locale archives are not used (see localedef(1)). The individual compiled locale data files are searched for under subdirectories which depend on

the currently used locale. For example, when en_GB.UTF-8 is used for a category, the following subdirectories are searched for, in this order: en_GB.UTF-8, en_GB.utf8, en_GB, en.UTF-8, en.utf8, and en.

FILES

/usr/lib/locale/locale-archive

Usual default locale archive location.

/usr/lib/locale

Usual default path for compiled individual locale files.

CONFORMING TO

POSIX.1-2001.

SEE ALSO

iconv(1), locale(1), localedef(1), catopen(3), gettext(3), iconv(3), localeconv(3), mbstowcs(3), newlocale(3), ngettext(3), nl_langinfo(3), rpmatch(3), setlocale(3), strcoll(3), strfmon(3), strftime(3), strxfrm(3), uselocale(3), wcstombs(3), locale(5), charsets(7), uni? code(7), utf-8(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/.