From peren!uunet!dkuug.dk!SC22WG14-request Sun Jul 24 06:13:05 1994
>Received: from dkuug.dk by relay1.UU.NET with SMTP
        id QQwzwf10625; Sat, 23 Jul 1994 21:20:42 -0400
X-Sequence: SC22WG14@dkuug.dk 622
X-Charset: ASCII
X-Char-Esc: 29
Mime-Version: 1.0
Content-Transfer-Encoding: 8bit
Mnemonic-Intro: 29
X-Mailer: Mail User's Shell (7.2.2 4/12/91)
To: uunet!dkuug.dk!wg14
Subject: (SC22WG14.622) C and POSIX
Content-Type: Text/Plain; Charset=ISO-8859-1
Content-Length: 8217
X-Lines: 242

WG14/N371
XJ11/94-056

Title: internationalization in C and POSIX standards

Date: 1994-07-22

Source: Keld Simonsen

Status: expert contribution

Distribution: ISO/IEC JTC1/SC22/WG14

references: earlier WG14 action item

Action: for discussion at WG14 Tokyo meeting

1. Introduction

This paper gives an overview of internationalisation in C and POSIX standards, and a comparison of the functionality and features provided. Also other incompatibilities between C and POSIX standards are mentioned. The covered standards are ISO/IEC 9899:1990 Programming Language C, ISO/IEC 9945-1:1990 POSIX System API (C language) (POSIX.1), and ISO/IEC 9945-2:1993 POSIX Shell and Utilities (POSIX.2). Internationalization may be abbreviated as I18N in the following.

2. The C standard

The C standard defines a 'locale' structure which holds data relevant to internationalization and a number of functions (denoted by func()), referencing/modifying values of the locale but no specification format to specify the values in the locales.

The following overview of C I18N functionality is rather terse, as it is considered well known to WG14 experts. It only references where I18N functionality is defined in the C standard, and major aspects of it.

C 7.3.1 character testing (using C operators)

isalnum	isaplha|isdigit
isalpha	isupper|islower| !(iscntrl|isdigit|ispunct|isspace)
iscntrl	
isdigit	'0'..'9'
isgraph	isprint | !SPACE
islower	'a'..'z' | !(iscntrl|isdigit|ispunct|isspace)
isprint	
ispunct	isprint & ! (SPACE | isalnum)
isspace	SPACE | \f | \n | \r | \t | \v | !isalnum
isupper	'A'..'Z' | !(iscntrl|isdigit|ispunct|isspace)
isxdigit	'0'..'9' | 'a'..'f' | 'A'..'F'

Remark: isupper and islower defines all nonspecial characters as both upper and lower, that is, all non-ASCII upper case letters are defined as lower, and all non-ASCII lowercase letters are defined as upper.

C 7.3.2 character case mapping

tolower for isupper, returns corresponding lower
toupper for islower, returns corresponding upper

Remark: no statement on locale dependence, nor how the correspondance is defined

C 7.4 localization

The C standard defines the following components of struct lconv:

```
char *decimal_point;
char *thousands_point;
char *grouping; CHAR_MAX 0 ? *
char *int_curr_symbol;
char *currency_symbol;
char *mon_decimal_point;
char *mon_thousands_sep;
char *mon_grouping; CHAR_MAX 0 ? *
char *positive_sign;
char *negative_sign;
char int_frac_digits;
char frac_digits;
char p_cs_precedes; 0 1
char p_sep_by_space; 0 1
char n_cs_precedes; 0 1
char n_sep_by_space; 0 1
char p_sign_posn; 0 1 2 3 4
char n_sign_posn; 0 1 2 3 4
```

C also defines the following macros:

```
NULL
LC_ALL
LC_COLLATE
LC_CTYPE
LC_MONETARY
LC_NUMERIC
LC_TIME
```

C 7.4.1.1 setlocale() uses above macroes

Remark: no reference to standardized locales, except the "C" locale.
A reference to registered locales of the forthcoming CEN standard are desirable.

C 7.4.2 localeconv() works on lconv values as noted above.

collating:
C 7.11.4.3 strcoll() and C 7.11.4.5 strxfrm() are dependent on the current locale's LC_COLLATE specification.

C 7.12.3.5 strftime() defines a number of formats:
a A b B c d H I j m M p S U w W x X y Y Z

C has various formatted I/O functions like fprintf() and fscanf() with formats dependent on the locale, described in C 7.9.6.

string conversion C 7.10.1 atof() and strtod() converts a string to double floating point representation, dependent on lconv values. strtol() and strtoul() are dependent on isspace().

3. POSIX.1

POSIX.1 defines the kernel interface, given in C language binding. For a lot of functionality, it does not define the I18N functionality, but relies on the C standard, which is included normatively with the C binding option of the standard. There is a separate setion (8) in POSIX.1 giving the extensions defined by POSIX.1 in relation to the C standard.

The extensions cover the following functions: setlocale, rename, getenv, ctime, gmtime, localtime, mktime and strftime. Also fseek and exit are specified furth

Extensions to the time functions concern the use of the environment variable TZ, to override system defaults.

A number of operating system considerations is done for various C I/O functions.

Please see POSIX.1 section 8 for further information.

4. POSIX.2

POSIX.2 defines a number of utilities, and very few functions. What is relevant for C in the alignment with POSIX standards, is also the specification format for the locale values, that is the localedef utility. Another area of interest is the date utility, where more formats are specified.

The new functionality in POSIX.2 adds a new category LC_MESSAGES to setlocale() in B.11. This is the only change of functionality.

In 2.5.2 a data description format for locale data is described. It corresponds closely to the functionality of C. The following lists changes:

A class "blank" is added, consisting initially of the characters <space> and <tab>.

All strings values of the monetary/numeric specifications can be with multiple characters (maybe a change to decimal_point, thousands_sep, mon_decimal_point and mon_thousands_sep is needed).

-1 is used instead of MAX_CHAR.

p_sep_by_space and n_sep_by_space have an additional value 2: if a space separates the symbol and the sign string, if adjacent.

p_sign_posn and n_sign_posn are also applicable to int_curr_symbol.

Extra data is defined for 'era' related information.

Also a new category LC_MESSAGES is defined with two strings yesexpr and noexpr, it is further meant to invoke the right messages corresponding to a locale.

A way of specifying the coded character set is defined in 2.4, but with no corresponding functions. It is used to compile localedef specifications into the relevant coded character set.

The date utility 4.15 has all of the formats C strftime() plus the following:

%C Century (a year divided by 100 and truncated to an integer) as a decimal number (00-99)
%D Date in the format mm/dd/yy
%e Day of the month as a decimal number (1-31 in a two-digit field with leading <space> fill)
%h a synonym for %b
%n a <newline> character
%r 12 h clock time (01-12) using the AM/PM notation; in the POSIX locale,

this shall be equivalent to "%I:%M:%S %p"
%t a <tab> character
%T 24 h clock time (00-23) in the format HH:MM:SS.
%u Week of the year (Sunday as the first day of the week) as a decimal number (00-53). All days in a new year preceding the first Sunday shall be considered to be in week 0.
%V Week of the year (Monday as the first day of the week) as a decimal number (00-53). The method for determining the week number shall be as specified in ISO 8601.

A number of modified field descriptors %O<d> and %E<d> are also defined (4.15.4.2).

5. POSIX.2b

An amendment to POSIX.2 is currently underway providing further specifications in the i18n area. This may be relevant to C functionality.

6. Recommendations.

To align the C standard with current POSIX standards, I propose the following:

6.1 Include the new formats and modifications of the POSIX.2 date utility in strftime()

6.2 Include the LC_MESSAGES category and the strings yesexpr and noexpr.

6.3 Include the new value for p_sep_by_space and n_sep_by_space.

6.4 Not make the extension to cover int_curr_symbol - but create a new series of values to handle int_curr_symbol.

6.5 State that all strings can have multiple characters (decimal_point, thousands_sep, mon_decimal_point and mon_thousands_sep).

6.6 Consider to use -1 instead of MAX_CHAR for various lconv values.

6.7 Add a function isblank(), corresponding to the POSIX.2 category "blank".

6.8 Reference POSIX.2 localedef for normative definition of data for locale and charmap.

6.9 Change isupper and islower to be able to not allow upper case letters as lowercase, and vice versa.

6.10 Consider referencing an international locale and charmap registry, maybe by a separate function.

6.11 Consider POSIX.1 section 8 (12 pages) for technical corrigenda, or for inclusion in amendment/revision of the C standard.

6.12 Watch out for POSIX.2b and align with this where possible.

<END OF DOCUMENT>
--
Keld Simonsen
Keld@dkuug.dk