IEEE P1003.2 Draft 11.2 – September 1991

IEEE Standards Department
Copyright and Permissions
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA
+1 (908) 562-3800
+1 (908) 562-1571 [FAX]

2.5.2.2.5 `order_end` Keyword

The collating order entries shall be terminated with an `order_end` keyword.

2.5.2.2.6 LC_COLLATE Rationale. *(This subclause is not a part of P1003.2)*

The LC_COLLATE category governs the collation order in the locale, and thus the processing of the C Standard {7} *strxfrm* () and *strcoll* () functions, as well as a number of POSIX. 2 utilities.

The rules governing collation depends to some extent on the use. At least five different levels of increasingly complex collation rules can be distinguished:

(1) Byte/machine code order. This is the historical collation order in the UNIX system and many proprietary operating systems. Collation is here done character by character, without any regard to context. The primary virtue is that it usually is quite fast, and also completely deterministic; it works well when the native machine collation sequence matches the user expectations.

(2) Character order. On this level, collation is also done character by character, without regard to context. The order between characters is, however, not determined by the code values, but on the user's expectations of the ''correct'' order between characters. In addition, such a (simple) collation order can specify that certain characters collate equal (e.g., upper- and lowercase letters).

(3) String ordering. On this level, entire strings are compared based on relatively straightforward rules. At this level, several ''passes'' may be required to determine the order between two strings. Characters may be ignored in some passes, but not in others; the strings may be compared in different directions; and simple string substitutions may be made before strings are compared. This level is best described as ''dictionary'' ordering; it is based on the spelling, not the pronunciation, or meaning, of the words.

(4) Text search ordering. This is a further refinement of the previous level,
best described as ''telephone book ordering''; some common homonyms 1
(words spelled differently but with same pronunciation) are collated 1
together; numbers are collated as if spelled with words, and so on.

(5) Semantic level ordering. Words and strings are collated based on their meaning; entire words (such as ''the'') are eliminated, the ordering is not deterministic. This usually requires special software, and is highly dependent on the intended use.

While the historical collation order formally is at level 1, for the English language it corresponds roughly to elements at level 2. The user expects to see the output from the `ls` utility sorted very much as as it would be in a dictionary. While telephone book ordering would be an optimal goal for standard collation, this was ruled out as the order would be language dependent. Furthermore, a requirement was that the order must be determined solely from the text string and the collation rules; no external information (e.g., ‘‘pronunciation dictionaries’’) could be required.

As a result, the goal for the collation support is at level 3. This also matches the requirements for the proposed Canadian collation order, as well as other, known collation requirements for alphabetic scripts. It specifically rules out collation based on pronunciation rules, or based on semantic analysis of the text.

The syntax for the LC_COLLATE category source is the result of a cooperative effort between representatives for many countries and organizations working with international issues, such as UniForum, X/Open, and ISO, and it meets the requirements for level 3, and has been verified to produce the correct result with examples based on French, Canadian, and Danish collation order, as well as meeting the requirements in the X/Open Portability Guide, Issue 3. {B31}. Because it supports multicharacter collating elements, it is also capable of supporting collation in code sets where a character is expressed using nonspacing characters followed by the base character (such as ISO 6937 {B6}).

The directives that can be specified in an operand to the `order_start` keyword 2
are based on the requirements specified in several proposed standards and in cus- 2
tomary use. The following is a rephrasing of rules defined for ‘‘lexical ordering in 2
English and French’’ by the Canadian Standards Association (text is brackets is 2
rephrased): 2

(1) Once special characters ([punctuation]) have been removed from original 2
strings, the ordering is determined by scanning forward (left to right) 2
[disregarding case and diacriticals]. 2

(2) In case of equivalence, special characters are once again removed from 2
original strings and the ordering is determined scanning backward 2
(starting from the rightmost character of the string and back), character 2
by character, [disregarding case but considering diacriticals]. 2

(3) In case of repeated equivalence, special characters are removed again 2
from original strings and the ordering is determined scanning forward, 2
character by character, [considering both case and diacriticals]. 2

(4) If there is still an ordering equivalence after rules (1) through (3) have 2
been applied, then only special characters and the position they occupy in 2
the string are considered to determine ordering. The string that has a 2
special character in the lowest position comes first. If two strings have a 2
special character in the same position, the character [with the lowest col- 2
lation value] comes first. In case of equality, the other special characters 2
are considered until there is a difference or all special characters have 2
been exhausted. 2

It is estimated that the standard covers the requirements for all European languages, and no particular problems are anticipated with Slavic or Middle East character sets.

The Far East (particularly Japanese/Chinese) collations are often based on contextual information and pronunciation rules (the same ideogram can have different meanings and different pronunciations). Such collation, in general, falls outside the desired goal of the standard. There are, however, several other collation rules (stroke/radical, or ‘‘most common pronunciation’’) which can be supported with the mechanism described here.

Previous drafts contained a `substitute` statement, which performed a regular 2
expression style replacement before string compares. It has been withdrawn 2
based on balloter objections that it was not required for the types of ordering 2
POSIX. 2 is aimed at. 2

The character (and collating element) order is defined by the order in which char- 2
acters and elements are specified between the `order_start` and `order_end` key- 2
words. This character order is used in range expressions in regular expressions 2
(see 2.8). Weights assigned to the characters and elements defines the collation 2
sequence; in the absence of weights, the character order is also the collation 2
sequence. 2

The `position` keyword was introduced to provide the capability to consider, in a 1
compare, the relative position of non-`IGNORE`d characters. As an example, con- 1
sider the two strings ‘‘o-ring’’ and ‘‘or-ing’’. Assuming the hyphen is `IGNORE`d on 1
the first pass, the two strings will compare equal, and the position of the hyphen 1
is immaterial. On second pass, all characters except the hyphen are `IGNORE`d, and 1
in the normal case the two strings would again compare equal. By taking position 1
into account, the first collates before the second. 1

2.5.2.3 LC_MONETARY

The LC_MONETARY category shall define the rules and symbols that shall be used to format monetary numeric information. The operands are strings. For some
keywords, the strings can contain only integers. Keywords that are not provided, 1
string values set to the empty string (`""`), or integer keywords set to −1, shall be 1
used to indicate that the value is unspecified. The following keywords shall be recognized:

`copy` Specify the name of an existing locale to be used as the source for the definition of this category. If this keyword is specified, no other keyword shall be specified.

`int_curr_symbol` The international currency symbol. The operand shall be a four-character string, with the first three characters containing the alphabetic international currency symbol in accordance with those specified in ISO 4217 {3} (*Codes for the representation of currencies and funds*). The fourth character shall be the character used to separate the international currency symbol from the monetary quantity.

`currency_symbol` The string that shall be used as the local currency symbol.

`mon_decimal_point` The operand is a string containing the symbol that 2
shall be used as the decimal delimiter in monetary for- 2
matted quantities. In contexts where other standards 2

Table 2-8 – LC_MONETARY Category Definition in the POSIX Locale

```
LC_MONETARY
# This is the POSIX Locale definition for
# the LC_MONETARY category.
#
int_curr_symbol     ""
currency_symbol     ""
mon_decimal_point   ""
mon_thousands_sep   ""
mon_grouping        ""
positive_sign       ""
negative_sign       ""
int_frac_digits     -1
p_cs_precedes       -1
p_sep_by_space      -1
n_cs_precedes       -1
n_sep_by_space      -1
p_sign_posn         -1
n_sign_posn         -1
#
END LC_MONETARY
```

limit the `mon_decimal_point` to a single byte, the result of specifying a multibyte operand is unspecified.

`mon_thousands_sep` The operand is a string containing the symbol that shall be used as a separator for groups of digits to the left of the decimal delimiter in formatted monetary quantities. In contexts where other standards limit the `mon_thousands_sep` to a single byte, the result of specifying a multibyte operand is unspecified.

`mon_grouping` Define the size of each group of digits in formatted monetary quantities. The operand is a sequence of integers separated by semicolons. Each integer specifies the number of digits in each group, with the initial integer defining the size of the group immediately preceding the decimal delimiter, and the following integers defining the preceding groups. If the last integer is not −1, then the size of the previous group (if any) shall be repeatedly used for the remainder of the digits. If the last integer is −1, then no further grouping shall be performed.

`positive_sign` A string that shall be used to indicate a nonnegative-valued formatted monetary quantity.

`negative_sign` A string that shall be used to indicate a negative-valued formatted monetary quantity.

`int_frac_digits` An integer representing the number of fractional digits (those to the right of the decimal delimiter) to be written in a formatted monetary quantity using `int_curr_symbol`.

`frac_digits` An integer representing the number of fractional digits (those to the right of the decimal delimiter) to be written in a formatted monetary quantity using `currency_symbol`.

`p_cs_precedes` An integer set to 1 if the `currency_symbol` or `int_curr_symbol` precedes the value for a nonnegative formatted monetary quantity, and set to 0 if the symbol succeeds the value.

`p_sep_by_space` An integer set to 0 if no space separates the `currency_symbol` or `int_curr_symbol` from the value for a nonnegative formatted monetary quantity, set to 1 if a space separates the symbol from the value, and set to 2 if a space separates the symbol and the sign string, if adjacent.

`n_cs_precedes` An integer set to 1 if the `currency_symbol` or `int_curr_symbol` precedes the value for a negative formatted monetary quantity, and set to 0 if the symbol succeeds the value.

`n_sep_by_space` An integer set to 0 if no space separates the `currency_symbol` or `int_curr_symbol` from the value for a negative formatted monetary quantity, set to 1 if a space separates the symbol from the value, and set to 2 if a space separates the symbol and the sign string, if adjacent.

`p_sign_posn` An integer set to a value indicating the positioning of the `positive_sign` for a nonnegative formatted monetary quantity. The following integer values shall be recognized:

- `0` Parentheses enclose the quantity and the `currency_symbol` or `int_curr_symbol`.
- `1` The sign string precedes the quantity and the `currency_symbol` or `int_curr_symbol`.
- `2` The sign string succeeds the quantity and the `currency_symbol` or `int_curr_symbol`.
- `3` The sign string immediately precedes the `currency_symbol` or `int_curr_symbol`.
- `4` The sign string immediately succeeds the `currency_symbol` or `int_curr_symbol`.

`n_sign_posn` An integer set to a value indicating the positioning of the `negative_sign` for a negative formatted mone- 1
tary quantity. The following integer values shall be recognized:

- `0` Parentheses enclose the quantity and the `currency_symbol` or `int_curr_symbol`.
- `1` The sign string precedes the quantity and the `currency_symbol` or `int_curr_symbol`.
- `2` The sign string succeeds the quantity and the `currency_symbol` or `int_curr_symbol`.
- `3` The sign string immediately precedes the `currency_symbol` or `int_curr_symbol`.
- `4` The sign string immediately succeeds the `currency_symbol` or `int_curr_symbol`.

2.5.2.3.1 LC_MONETARY Rationale. *(This subclause is not a part of P1003.2)*

The currency symbol does not appear in LC_MONETARY because it is not defined in the C Standard's {7} C locale.

The C Standard {7} limits the size of decimal points and thousands delimiters to 2
single-byte values. In locales based on multibyte coded character sets this cannot 2
be enforced, obviously; this standard does not prohibit such characters, but makes 2
the behavior unspecified [in the text ''In contexts where other standards ...'']. 2

The grouping specification is based on, but not identical to, the C Standard {7}. 2
The ''−1'' signals that no further grouping shall be performed, the equivalent of 2
{CHAR_MAX} in the C Standard {7}). 2

The locale definition is an extension of the C Standard {7} *localeconv* () specification. In particular, rules on how `currency_symbol` is treated are extended to also cover `int_curr_symbol`, and `p_set_by_space` and `n_sep_by_space` have been augmented with the value 2, which places a space between the sign and the symbol (if they are adjacent; otherwise it should be treated as a 0). The following table shows the result of various combinations:

		p_sep_by_space 2	1	0
p_cs_precedes = 1	p_sign_posn = 0	($1.25)	($ 1.25)	($1.25)
	p_sign_posn = 1	+ $1.25	+$ 1.25	+$1.25
	p_sign_posn = 2	$1.25 +	$ 1.25+	$1.25+
	p_sign_posn = 3	+ $1.25	+$ 1.25	+$1.25
	p_sign_posn = 4	$ +1.25	$+ 1.25	$+1.25
p_cs_precedes = 0	p_sign_posn = 0	(1.25 $)	(1.25 $)	(1.25$)
	p_sign_posn = 1	+1.25 $	+1.25 $	+1.25$
	p_sign_posn = 2	1.25$ +	1.25 $+	1.25$+
	p_sign_posn = 3	1.25+ $	1.25 +$	1.25+$
	p_sign_posn = 4	1.25$ +	1.25 $+	1.25$+

The following is an example of the interpretation of the mon_grouping keyword.
Assuming that the value to be formatted is 123456789 and the
mon_thousands_sep is ', then the following table shows the result. The third 1
column shows the equivalent C Standard {7} string that would be used to accom- 1
modate this grouping. It is the responsibility of the utility to perform mappings of 1
the formats in this clause to those used by language bindings such as the 1
C Standard {7}. 1

mon_grouping	Formatted Value	C Standard {7} String	
3;-1	123456'789	"\3\177"	2
3	123'456'789	"\3"	2
3;2;-1	1234'56'789	"\3\2\177"	2
3;2	12'34'56'789	"\3\2"	2
-1	123456789	"177"	2

In these examples, the octal value of {CHAR_MAX} is 177. 2

2.5.2.4 LC_NUMERIC

The LC_NUMERIC category shall define the rules and symbols that shall be used
to format nonmonetary numeric information. The operands are strings. For some
keywords, the strings only can contain integers. Keywords that are not provided, 1
string values set to the empty string (""), or integer keywords set to −1, shall be 1
used to indicate that the value is unspecified. The following keywords shall be
recognized:

copy Specify the name of an existing locale to be used as the source for the definition of this category. If this keyword is specified, no other keyword shall be specified.

decimal_point The operand is a string containing the symbol that shall 2
be used as the decimal delimiter in numeric, nonmonetary 2
formatted quantities. This keyword cannot be omitted 2
and cannot be set to the empty string. In contexts where 2
other standards limit the decimal_point to a single 2
byte, the result of specifying a multibyte operand is 2

unspecified.

`thousands_sep` The operand is a string containing the symbol that shall be used as a separator for groups of digits to the left of the decimal delimiter in numeric, nonmonetary formatted monetary quantities. In contexts where other standards limit the `thousands_sep` to a single byte, the result of specifying a multibyte operand is unspecified.

`grouping` Define the size of each group of digits in formatted non-monetary quantities. The operand is a sequence of integers separated by semicolons. Each integer specifies the number of digits in each group, with the initial integer defining the size of the group immediately preceding the decimal delimiter, and the following integers defining the preceding groups. If the last integer is not −1, then the size of the previous group (if any) shall be repeatedly used for the remainder of the digits. If the last integer is −1, then no further grouping shall be performed.

Table 2-9 – LC_NUMERIC Category Definition in the POSIX Locale

```
LC_NUMERIC
# This is the POSIX Locale definition for
# the LC_NUMERIC category.
#
decimal_point       "<period>"
thousands_sep       ""
grouping            0
#
END LC_NUMERIC
```

2.5.2.4.1 LC_NUMERIC Rationale. *(This subclause is not a part of P1003.2)*

See the rationale for LC_MONETARY (2.5.2.3.1) for a description of the behavior of grouping.

2.5.2.5 LC_TIME

The LC_TIME category shall define the interpretation of the field descriptors supported by the `date` utility (see 4.15).

The following mandatory keywords shall be recognized:

`copy` Specify the name of an existing locale to be used as the source for the definition of this category. If this keyword is specified, no other keyword shall be specified.

`abday` Define the abbreviated weekday names, corresponding to the `%a` field descriptor. The operand shall consist of seven semicolon-separated strings. The first string shall be the abbreviated name

Table 2-10 – LC_TIME Category Definition in the POSIX Locale

```
LC_TIME
# This is the POSIX Locale definition for
# the LC_TIME category.
#
# Abbreviated weekday names (%a)
abday   "<S><u><n>";"<M><o><n>";"<T><u><e>";"<W><e><d>";\
        "<T><h><u>";"<F><r><i>";"<S><a><t>"
#
# Full weekday names (%A)
day     "<S><u><n><d><a><y>";"<M><o><n><d><a><y>";\
        "<T><u><e><s><d><a><y>";"<W><e><d><n><e><s><d><a><y>";\
        "<T><h><u><r><s><d><a><y>";"<F><r><i><d><a><y>";\
        "<S><a><t><u><r><d><a><y>"
#
# Abbreviated month names (%b)
abmon   "<J><a><n>";"<F><e><b>";"<M><a><r>";\
        "<A><p><r>";"<M><a><y>";"<J><u><n>";\
        "<J><u><l>";"<A><u><g>";"<S><e><p>";\
        "<O><c><t>";"<N><o><v>";"<D><e><c>"
#
# Full month names (%B)
mon     "<J><a><n><u><a><r><y>";"<F><e><b><r><u><a><r><y>";\
        "<M><a><r><c><h>";"<A><p><r><i><l>";\
        "<M><a><y>";"<J><u><n><e>";\
        "<J><u><l><y>";"<A><u><g><u><s><t>";\
        "<S><e><p><t><e><m><b><e><r>";"<O><c><t><o><b><e><r>";\
        "<N><o><v><e><m><b><e><r>";"<D><e><c><e><m><b><e><r>"
#
# Equivalent of AM/PM (%p)      "AM";"PM"
am_pm   "<A><M>";"<P><M>"
#
# Appropriate date and time representation (%c)
#       "%a %b %e %H:%M:%S %Y"                                              1
d_t_fmt "<percent-sign><a><space><percent-sign><b><space><percent-sign><e>\   1
<space><percent-sign><H><colon><percent-sign><M>\
<colon><percent-sign><S><space><percent-sign><Y>"
#
# Appropriate date representation (%x)   "%m/%d/%y"
d_fmt   "<percent-sign><m><slash><percent-sign><d><slash><percent-sign><y>"
#
# Appropriate time representation (%X)   "%H:%M:%S"
t_fmt   "<percent-sign><H><colon><percent-sign><M><colon><percent-sign><S>"
#
# Appropriate 12-hour time representation (%r) "%I:%M:%S %p"
t_fmt_ampm "<percent-sign><I><colon><percent-sign><M><colon>\
<percent-sign><S> <percent_sign><p>"
#
END LC_TIME
```

of the first day of the week (Sunday), the second the abbreviated name of the second day, and so on.

`day` Define the full weekday names, corresponding to the `%A` field descriptor. The operand shall consist of seven semicolon-separated strings. The first string shall be the full name of the first day of the week (Sunday), the second the full name of the second day, and so on.

`abmon` Define the abbreviated month names, corresponding to the `%b` field descriptor. The operand shall consist of twelve semicolon-separated strings. The first string shall be the abbreviated name of the first month of the year (January), the second the abbreviated name of the second month, and so on.

`mon` Define the full month names, corresponding to the `%B` field descriptor. The operand shall consist of twelve semicolon-separated strings. The first string shall be the full name of the first month of the year (January), the second the full name of the second month, and so on.

`d_t_fmt` Define the appropriate date and time representation, corresponding to the `%c` field descriptor. The operand shall consist of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain escape sequences defined
in Table 2-15. 1

`d_fmt` Define the appropriate date representation, corresponding to the `%x` field descriptor. The operand shall consist of a string, and can contain any combination of characters and field descriptors. In
addition, the string can contain escape sequences defined in 1
Table 2-15. 1

`t_fmt` Define the appropriate time representation, corresponding to the `%X` field descriptor. The operand shall consist of a string, and can contain any combination of characters and field descriptors. In
addition, the string can contain escape sequences defined in 1
Table 2-15. 1

`am_pm` Define the appropriate representation of the *ante meridiem* and *post meridiem* strings, corresponding to the `%p` field descriptor. The operand shall consist of two strings, separated by a semicolon. The first string shall represent the *ante meridiem* designation, the last string the *post meridiem* designation.

`t_fmt_ampm`

Define the appropriate time representation in the 12-hour clock format with `am_pm`, corresponding to the `%r` field descriptor. The operand shall consist of a string and can contain any combination of characters and field descriptors. If the string is empty, the 12-hour format is not supported in the locale.

It is implementation defined whether the following optional keywords shall be recognized. If they are not supported, but present in a `localedef` source, they shall be ignored.

`era` Shall be used to define alternate Eras, corresponding to the `%E` field descriptor modifier. The format of the operand is unspecified, but shall support the definition of the `%EC` and `%Ey` field descriptors, and may also define the `era_year` format (`%EY`).

`era_year` Shall be used to define the format of the year in alternate Era format, corresponding to the `%EY` field descriptor.

`era_d_fmt` Shall be used to define the format of the date in alternate Era notation, corresponding to the `%Ex` field descriptor.

`alt_digits` Shall be used to define alternate symbols for digits, corresponding to the `%O` field descriptor modifier. The operand shall consist of semicolon-separated strings. The first string shall be the alternate symbol corresponding with zero, the second string the symbol corresponding with one, and so on. Up to 100 alternate symbol strings can be specified. The `%O` modifier indicates that the string corresponding to the value specified via the field descriptor shall be used instead of the value.

2.5.2.5.1 LC_TIME Rationale. *(This subclause is not a part of P1003.2)*

Although certain of the field descriptors in the POSIX Locale (such as the name of the month) are shown with initial capital letters, this need not be the case in other locales. Programs using these fields may need to adjust the capitalization if the output is going to be used at the beginning of a sentence.

The LC_TIME descriptions of `abday`, `daya`, and `abmon` imply a Gregorian style 1
calendar (7-day weeks, 12-month years, leap years, etc.). Formatting time strings 1
for other types of calendars is outside the scope of this standard. 1

As specified under the `date` command, the field descriptors corresponding to the optional keywords consist of a modifier followed by a traditional field descriptor (for instance `%Ex`). If the optional keywords are not supported by the implementation or are unspecified for the current locale, these field descriptors shall be treated as the traditional field descriptor. For instance, assume the following keywords:

```
alt_digits      "0th";"1st";"2nd";"3rd";"4th";"5th";\        1
                "6th";"7th";"8th";"9th";"10th"               1

d_fmt           "The %Od day of %B in %Y"                    1
```

On 7/4/1776, the `%x` field descriptor would result in ‘‘The 4th day of July in 1776,’’ 1
while 7/14/1789 would come out as ‘‘The 14 day of July in 1789.’’ It can be noted that the above example is for illustrative purposes only; the `%O` modifier is primarily intended to provide for Kanji or Hindi digits in `date` formats.

While it is clear that an alternate year format is required, there is no consensus on the format or the requirements. As a result, while these keywords are reserved, the details are left unspecified. It is expected that National Standards Bodies will provide specifications.

2.5.2.6 LC_MESSAGES

The LC_MESSAGES category shall define the format and values for affirmative and negative responses. The operands shall be strings or extended regular expressions; see 2.8.4. The following keywords shall be recognized:

`copy` Specify the name of an existing locale to be used as the source for the definition of this category. If this keyword is specified, no other keyword shall be specified.

`yesexpr` The operand shall consist of an extended regular expression that describes the acceptable affirmative response to a question expecting an affirmative or negative response.

`noexpr` The operand shall consist of an extended regular expression that describes the acceptable negative response to a question expecting an affirmative or negative response.

Table 2-11 – LC_MESSAGES Category Definition in the POSIX Locale

```
LC_MESSAGES
# This is the POSIX Locale definition for
# the LC_MESSAGES category.
#
yesexpr "<circumflex><left-square-bracket><y><Y><right-square-bracket>"
#
noexpr  "<circumflex><left-square-bracket><n><N><right-square-bracket>"
END LC_MESSAGES
```

2.5.2.6.1 LC_MESSAGES Rationale. *(This subclause is not a part of P1003.2)*

The LC_MESSAGES category is described in 2.6 as affecting the language used by utilities for their output. The mechanism used by the implementation to accomplish this, other than the responses shown here in the locale definition file, is not specified by this version of this standard. The POSIX.1 working group is developing an interface that would allow applications (and, presumably some of the standard utilities) to access messages from various message catalogs, tailored to a user's **LC_MESSAGES** value.

2.5.3 Locale Definition Grammar

The grammar and lexical conventions in this subclause shall together describe the syntax for the locale definition source. The general conventions for this style of grammar are described in 2.1.2. Any discrepancies found between this grammar and other descriptions in this clause shall be resolved in favor of this grammar.

2.5.3.1 Locale Lexical Conventions

The lexical conventions for the locale definition grammar are described in this subclause.

The following tokens shall be processed (in addition to those string constants shown in the grammar):

`LOC_NAME` A string of characters representing the name of a locale.

`CHAR` Any single character.

`NUMBER` A decimal number, represented by one or more decimal digits.

`COLLSYMBOL` A symbolic name, enclosed between angle brackets. The string shall not duplicate any charmap symbol defined in the current charmap (if any), or a `COLLELEMENT` symbol.

`COLLELEMENT` A symbolic name, enclosed between angle brackets, which shall not duplicate either any charmap symbol or a `CHARSYMBOL` symbol.

`CHARSYMBOL` A symbolic name, enclosed between angle brackets, from the current charmap (if any).

`OCTAL_CHAR` One or more octal representations of the encoding of each byte in a single character. The octal representation consists of an `escape_char` (normally a backslash) followed by two or more octal digits.

`HEX_CHAR` One or more hexadecimal representations of the encoding of each byte in a single character. The hexadecimal representation consists of an `escape_char` followed by the constant `'x'` and two or more hexadecimal digits.

`DECIMAL_CHAR` One or more decimal representations of the encoding of each byte in a single character. The decimal representation consists of an `escape_char` and followed by a `'d'` and two or more decimal digits.

`ELLIPSIS` The string ``...''.

`EXTENDED_REG_EXP`
An extended regular expression as defined in the grammar in 2.8.5.2.

`EOL` The line termination character `<newline>`. 1

2.5.3.2 Locale Grammar 1

This subclause presents the grammar for the locale definition. 1

```
%token          LOC_NAME                                                   1
%token          CHAR                                                       1
%token          NUMBER                                                     2
%token          COLLSYMBOL COLLELEMENT                                     1
%token          CHARSYMBOL OCTAL_CHAR HEX_CHAR DECIMAL_CHAR                1
%token          ELLIPSIS                                                   1
%token          EXTENDED_REG_EXP                                           2
%token          EOL                                                        1

%start  locale_definition                                                  1

%%                                                                         1

locale_definition       : global_statements locale_categories              2
                        |                   locale_categories              2
                        ;                                                  1

global_statements       : global_statements symbol_redefine                2
                        | symbol_redefine                                  2
                        ;                                                  1

symbol_redefine         : '#escape_char'  CHAR EOL                         1
                        | '#comment_char' CHAR EOL                         1
                        ;                                                  1

locale_categories       : locale_categories locale_category                2
                        | locale_category                                  2
                        ;                                                  1

locale_category         : lc_ctype | lc_collate | lc_messages              1
                        | lc_monetary | lc_numeric | lc_time               1
                        ;                                                  1

/*      The following grammar rules are common to all categories */       1

char_list               : char_list char_symbol                            2
                        | char_symbol                                      2
                        ;                                                  1

char_symbol             : CHAR | CHARSYMBOL                                1
                        | OCTAL_CHAR | HEX_CHAR | DECIMAL_CHAR             1
                        ;                                                  1

locale_name             : LOC_NAME                                         1
                        | '"' LOC_NAME '"'                                 1
                        ;                                                  1

/*      The following is the LC_CTYPE category grammar */                 1

lc_ctype                : ctype_hdr ctype_keywords         ctype_tlr       2
                        | ctype_hdr 'copy' locale_name EOL ctype_tlr       2
                        ;                                                  2
```

```
ctype_hdr               : 'LC_CTYPE' EOL                                        2
                        ;                                                       2

ctype_keywords          : ctype_keywords ctype_keyword                          2
                        | ctype_keyword                                         2
                        ;                                                       1

ctype_keyword           : charclass_keyword charclass_list EOL                  1
                        | charconv_keyword charconv_list EOL                    1
                        ;                                                       1

charclass_keyword       : 'upper' | 'lower' | 'alpha' | 'digit'                 1
                        | 'alnum' | 'xdigit' | 'space' | 'print'                1
                        | 'graph' | 'blank' | 'cntrl'                           1
                        ;                                                       1

charclass_list          : charclass_list ';' char_symbol                        2
                        | charclass_list ';' ELLIPSIS ';' char_symbol           1
                        | char_symbol                                           2
                        ;                                                       1

charconv_keyword        : 'toupper'                                             1
                        | 'tolower'                                             1
                        ;                                                       1

charconv_list           : charconv_list ';' charconv_entry                      2
                        | charconv_entry                                        2
                        ;                                                       1

charconv_entry          : '(' char_symbol ',' char_symbol ')'                   1
                        ;                                                       1

ctype_tlr               : 'END' 'LC_CTYPE' EOL                                  2
                        ;                                                       1

/*      The following is the LC_COLLATE category grammar */                     1

lc_collate              : collate_hdr collate_keywords       collate_tlr        2
                        | collate_hdr 'copy' locale_name EOL collate_tlr        2
                        ;                                                       2

collate_hdr             : 'LC_COLLATE' EOL                                      2
                        ;                                                       2

collate_keywords        :                order_statements                       2
                        | opt_statements order_statements                       2
                        ;                                                       1

opt_statements          : opt_statements collating_symbols                      2
                        | opt_statements collating_elements                     2
                        | collating_symbols                                     1
                        | collating_elements                                    1
                        ;                                                       1

collating_symbols       : 'collating-symbol' COLLSYMBOL EOL                     1
                        ;                                                       1

collating_elements      : 'collating-element' COLLELEMENT                       1
                          'from' '"' char_list '"' EOL                          2
                        ;                                                       1
```

```
order_statements       : order_start collation_order order_end           1
                       ;                                                 1
order_start            : 'order_start' EOL                               1
                       | 'order_start' order_opts EOL                    1
                       ;                                                 1
order_opts             : order_opts ';' order_opt                        2
                       | order_opt                                       2
                       ;                                                 1
order_opt              : order_opt ',' opt_word                          2
                       | opt_word                                        2
                       ;                                                 1
opt_word               : 'forward' | 'backward' | 'position'             2
                       ;                                                 1
collation_order        : collation_order collation_entry                 2
                       | collation_entry                                 2
                       ;                                                 1
collation_entry        : COLLSYMBOL EOL                                  1
                       | collation_element weight_list EOL               1
                       | collation_element             EOL               2
                       ;                                                 1
collation_element      : char_symbol                                     1
                       | COLLELEMENT                                     1
                       | ELLIPSIS                                        1
                       | 'UNDEFINED'                                     1
                       ;                                                 1
weight_list            : weight_list ';' weight_symbol                   2
                       | weight_list ';'                                 2
                       | weight_symbol                                   2
                       ;                                                 1
weight_symbol          : char_symbol                                     2
                       | COLLSYMBOL                                      1
                       | '"' char_list '"'                               1
                       | ELLIPSIS                                        1
                       | 'IGNORE'                                        1
                       ;                                                 1
order_end              : 'order_end' EOL                                 1
                       ;                                                 1
collate_tlr            : 'END' 'LC_COLLATE' EOL                          2
                       ;                                                 1
/*      The following is the LC_MESSAGES category grammar */             1
lc_messages            : messages_hdr messages_keywords      messages_tlr    2
                       | messages_hdr 'copy' locale_name EOL messages_tlr    2
                       ;                                                 2
messages_hdr           : 'LC_MESSAGES' EOL                               2
                       ;                                                 2
```

```
messages_keywords       : messages_keywords messages_keyword            2
                        | messages_keyword                              2
                        ;                                               1

messages_keyword        : ’yesexpr’ ’"’ EXTENDED_REG_EXP ’"’ EOL        2
                        | ’noexpr’  ’"’ EXTENDED_REG_EXP ’"’ EOL        2
                        ;                                               2

messages_tlr            : ’END’ ’LC_MESSAGES’ EOL                       2
                        ;                                               1

/*      The following is the LC_MONETARY category grammar */            1

lc_monetary             : monetary_hdr monetary_keywords       monetary_tlr   2
                        | monetary_hdr ’copy’ locale_name EOL  monetary_tlr   2
                        ;                                               2

monetary_hdr            : ’LC_MONETARY’ EOL                             2
                        ;                                               2

monetary_keywords       : monetary_keywords monetary_keyword            2
                        | monetary_keyword                              2
                        ;                                               1

monetary_keyword        : mon_keyword_string mon_string EOL             1
                        | mon_keyword_char NUMBER EOL                   2
                        | mon_keyword_char ’-1’   EOL                   2
                        | mon_keyword_grouping mon_group_list EOL       1
                        ;                                               1

mon_keyword_string      : ’int_curr_symbol’ | ’currency_symbol’         1
                        | ’mon_decimal_point’ | ’mon_thousands_sep’     1
                        | ’positive_sign’ | ’negative_sign’             1
                        ;                                               1

mon_string              : ’"’ char_list ’"’                             1
                        | ’""’                                          1
                        ;                                               1

mon_keyword_char        : ’int_frac_digits’ | ’frac_digits’             1
                        | ’p_cs_precedes’ | ’p_sep_by_space’            1
                        | ’n_cs_precedes’ | ’n_sep_by_space’            1
                        | ’p_sign_posn’ | ’n_sign_posn’                 1
                        ;                                               1


mon_keyword_grouping    : ’mon_grouping’                                1
                        ;                                               1

mon_group_list          : NUMBER                                        2
                        | mon_group_list ’;’ NUMBER                     2
                        ;                                               2

monetary_tlr            : ’END’ ’LC_MONETARY’ EOL                       2
                        ;                                               2

/*      The following is the LC_NUMERIC category grammar */             2

lc_numeric              : numeric_hdr numeric_keywords       numeric_tlr      2
                        | numeric_hdr ’copy’ locale_name EOL numeric_tlr      2
                        ;                                               2
```

```
numeric_hdr             : 'LC_NUMERIC' EOL                                   2
                        ;                                                    2

numeric_keywords        : numeric_keywords numeric_keyword                   2
                        | numeric_keyword                                    2
                        ;                                                    1

numeric_keyword         : num_keyword_string num_string EOL                  1
                        | num_keyword_grouping num_group_list EOL            1
                        ;                                                    1

num_keyword_string      : 'decimal_point'                                    1
                        | 'thousands_sep'                                    1
                        ;                                                    1

num_string              : '"' char_list '"'                                  1
                        | '""'                                               1
                        ;                                                    1

num_keyword_grouping    : 'num_grouping'                                     1
                        ;                                                    1

num_group_list          : NUMBER                                             2
                        | num_group_list ';' NUMBER                          2
                        ;                                                    1


numeric_tlr             : 'END' 'LC_NUMERIC' EOL                             2
                        ;                                                    1

/*      The following is the LC_TIME category grammar */                     1

lc_time                 : time_hdr time_keywords          time_tlr           2
                        | time_hdr 'copy' locale_name EOL time_tlr           2
                        ;                                                    1

time_hdr                : 'LC_TIME' EOL                                      2
                        ;                                                    1

time_keywords           : time_keywords time_keyword                         2
                        | time_keyword                                       2
                        ;                                                    1

time_keyword            : time_keyword_name time_list EOL                    2
                        | time_keyword_fmt time_string EOL                   1
                        | time_keyword_opt time_list EOL                     1
                        ;                                                    1

time_keyword_name       : 'abday' | 'day' | 'abmon' | 'mon'                  2
                        ;                                                    1

time_keyword_fmt        : 'd_t_fmt' | 'd_fmt' | 't_fmt' | 'am_pm' | 't_fmt_ampm' 1
                        ;                                                    1

time_keyword_opt        : 'era' | 'era_year' | 'era_d_fmt' | 'alt_digits'    1
                        ;                                                    1

time_list               : time_list ';' time_string                          2
                        | time_string                                        2
                        ;                                                    1

time_string             : '"' char_list '"'                                  1
                        ;                                                    1
```

```
time_tlr                : 'END' 'LC_TIME' EOL                         2
                        ;                                             1
```

2.5.4 Locale Definition Example. *(This subclause is not a part of P1003.2)*

The following is an example of a locale definition file that could be used as input to the `localedef` utility. It assumes that the utility is executed with the `−f` option, naming a *charmap* file with (at least) the following content:

```
CHARMAP
<space>       \x20
<dollar>      \x24
<A>           \101
<a>           \141
<A-acute>     \346
<a-acute>     \365
<A-grave>     \300                                                    1
<a-grave>     \366
<b>           \142
<C>           \103
<c>           \143
<c-cedilla>   \347
<d>           \x64
<H>           \110
<h>           \150
<eszet>       \xb7
<s>           \x73
<z>           \x7a
END CHARMAP
```

It should not be taken as complete or to represent any actual locale, but only to illustrate the syntax.

A further set of examples is offered as part of Annex F.

```
#
LC_CTYPE
lower   <a>;<b>;<c>;<c-cedilla>;<d>;...;<z>
upper   A;B;C;Ç;...;Z
space   \x20;\x09;\x0a;\x0b;\x0c;\x0d                                 1
blank   \040;\011
toupper (<a>,<A>);(b,B);(c,C);(ç,Ç);(d,D);(z,Z)
END LC_CTYPE
#
LC_COLLATE
#
# The following example of collation is based on the proposed         1
# Canadian standard Z243.4.1-1990, "Canadian Alphanumeric              1
# Ordering Standard For Character sets of CSA Z234.4 Standard".        1
# (Other parts of this example locale definition file do not          1
# purport to relate to Canada, or to any other real culture.)         1
# The proposed standard defines a 4-weight collation, such that
# in the first pass, characters are compared without regard to
```

```
# case or accents; in second pass, backwards compare without
# regard to case; in the third pass, forward compare without
# regard to diacriticals.  In the 3 first passes, non-alphabetic      2
# characters are ignored; in the fourth pass, only special
# characters are considered, such that "The string that has a
# special character in the lowest position comes first.  If two
# strings have a special character in the same position, the
# collation value of the special character determines ordering.
#
# Only a subset of the character set is used here; mostly to
# illustrate the set-up.
#
                                                                      2
#
collating-symbol <LOW_VALUE>                                          2
collating-symbol <LOWER-CASE>
collating-symbol <SUBSCRIPT-LOWER>
collating-symbol <SUPERSCRIPT-LOWER>
collating-symbol <UPPER-CASE>
collating-symbol <NO-ACCENT>
collating-symbol <PECULIAR>
collating-symbol <LIGATURE>
collating-symbol <ACUTE>
collating-symbol <GRAVE>
# Further collating-symbols follow.
#
# Properly, the standard does not include any multi-character
# collating elements; the one below is added for completeness.
#
collating_element <ch> from <c><h>
collating_element <CH> from <C><H>
collating_element <Ch> from <C><h>
#
order_start forward;backward;forward;forward,position
#
# Collating symbols are specified first in the sequence to allocate
# basic collation values to them, lower that than of any character.

<LOW_VALUE>                                                           2
<LOWER-CASE>
<SUBSCRIPT-LOWER>
<SUPERSCRIPT-LOWER>
<UPPER-CASE>
<NO-ACCENT>
<PECULIAR>
<LIGATURE>
<ACUTE>
<GRAVE>
<RING-ABOVE>
<DIAERESIS>
<TILDE>
# Further collating symbols are given a basic collating value here.
```