IEEE P1003.2 Draft 11.2 – September 1991

Copyright © 1991 by the
Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA
All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change. The publication, distribution, or copying of this draft, as well as all derivative works based on this draft, is expressly prohibited except as set forth below.

Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only, and subject to the restrictions contained herein.

Permission is hereby also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position, subject to the restrictions contained herein.

Permission is hereby also granted to the preceding entities to make limited copies of this document in an electronic form only for the stated activities.

The following restrictions apply to reproducing or transmitting the document in any form: 1) all copies or portions thereof must identify the document's IEEE project number and draft number, and must be accompanied by this entire notice in a prominent location; 2) no portion of this document may be redistributed in any modified or abridged form without the prior approval of the IEEE Standards Department.

Other entities seeking permission to reproduce this document, or any portion thereof, for standardization or other activities, must contact the IEEE Standards Department for the appropriate license.

Use of information contained in this unapproved draft is at your own risk.

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]

This standard retains the `-f` option to provide support for shell scripts depending on the *SVID* semantics. It seems likely that shell scripts would not be written to handle prompting by `ln`, and would therefore have specified the `-f` option.

It should also be noted that `-f` is an undocumented feature of many historical versions of the `ln` utility, allowing linking to directories. These versions will require modification.

Previous drafts of this standard also required an `-i` option, which behaved like the `-i` options in `cp` and `mv`, prompting for confirmation before unlinking existing files. This was not historical practice for the `ln` utility and has been deleted from this version.

Although symbolic links are not part of the standard, the `-s` option should be used only for the traditional purpose of creating symbolic links.

4.34 `locale` — Get locale-specific information

4.34.1 Synopsis

```
locale [-a|-m]
locale [-ck] name ...
```

4.34.2 Description

The `locale` utility shall write information about the current locale environment, or all public locales, to the standard output. For the purposes of this clause, a *public locale* is one provided by the implementation that is accessible to the application.

When `locale` is invoked without any arguments, it shall summarize the current locale environment for each locale category as determined by the settings of the environment variables defined in 2.5.

When invoked with operands, it shall write values that have been assigned to the keywords in the locale categories, as follows:

— Specifying a keyword name shall select the named keyword and the category containing that keyword.

— Specifying a category name shall select the named category and all keywords in that category.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

4.34.3 Options

The `locale` utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following options shall be supported by the implementation:

`-a` Write information about all available public locales. The available locales shall include `POSIX`, representing the POSIX Locale. The manner in which the implementation determines what other locales are available is implementation defined.

`-c` Write the names of selected locale categories; see 4.34.6.1.

`-k` Write the names and values of selected keywords. The implementation may omit values for some keywords; see 4.34.4.

`-m` Write names of available charmaps; see 2.4.1.

4.34.4 Operands

The following operand shall be supported by the implementation:

name The name of a locale category as defined in 2.5, the name of a keyword in a locale category, or the reserved name `charmap`. The named category or keyword shall be selected for output. If a single *name* represents both a locale category name and a keyword name in the current locale, the results are unspecified. Otherwise, both category and keyword names can be specified as *name* operands, in any sequence. It is implementation defined whether any keyword values are written for the categories LC_CTYPE and LC_COLLATE.

4.34.5 External Influences

4.34.5.1 Standard Input

None.

4.34.5.2 Input Files

None.

4.34.5.3 Environment Variables

The following environment variables shall affect the execution of `locale`:

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

7653 **LANG** This variable shall determine the locale to use for the
7654 locale categories when both **LC_ALL** and the correspond-
7655 ing environment variable (beginning with **LC_**) do not
7656 specify a locale. See 2.6.

7657 **LC_ALL** This variable shall determine the locale to be used to over-
7658 ride any values for locale categories specified by the set-
7659 tings of **LANG** or any environment variables beginning
7660 with **LC_**.

7661 **LC_CTYPE** This variable shall determine the locale for the interpreta-
7662 tion of sequences of bytes of text data as characters (e.g.,
7663 single- versus multibyte characters in arguments).

7664 **LC_MESSAGES** This variable shall determine the language in which mes-
7665 sages should be written.

7666 The **LANG** and **LC_*** environment variables shall specify the current locale
7667 environment to be written out; they shall be used if the `-a` option is not specified.

7668 #### 4.34.5.4 Asynchronous Events

7669 Default.

7670 ### 4.34.6 External Effects

7671 #### 4.34.6.1 Standard Output

7672 If `locale` is invoked without any options or operands, the names and values of
7673 the **LANG** and **LC_*** environment variables described in this standard shall be
7674 written to the standard output, one variable per line, with **LANG** first, and each
7675 line using the following format. Only those variables set in the environment and
7676 not overridden by **LC_ALL** shall be written using this format:

7677 `"%s=%s\n"`, *<variable_name>* , *<value>*

7678 The names of those **LC_*** variables associated with locale categories defined in
7679 this standard that are not set in the environment or are overridden by **LC_ALL**
7680 shall be written in the following format:

7681 `"%s=\"%s\"\n"`, *<variable_name>* , *<implied value>*

7682 The *<implied value>* shall be the name of the locale that has been selected for
7683 that category by the implementation, based on the values in **LANG** and **LC_ALL**,
7684 as described in 2.6.

7685 The *<value>* and *<implied value>* shown above shall be properly quoted for possi- 1
7686 ble later re-entry to the shell. The *<value>* shall not be quoted using double- 1
7687 quotes (so that it can be distinguished by the user from the *<implied value>* case, 1
7688 which always requires double-quotes). 1

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

The **LC_ALL** variable shall be written last, using the first format shown above. If it is not set, it shall be written as:

```
"LC_ALL=\n"
```

If any arguments are specified:

(1) If the −a option is specified, the names of all the public locales shall be written, each in the following format:

"%s\n", <*locale name*>

(2) If the −c option is specified, the name(s) of all selected categories shall be written, each in the following format:

"%s\n", <*category name*>

If keywords are also selected for writing (see following items), the category name output shall precede the keyword output for that category.

If the −c option is not specified, the names of the categories shall not be written; only the keywords, as selected by the *name* operand, shall be written.

(3) If the −k option is specified, the name(s) and value(s) of selected keywords shall be written. If a value is nonnumeric, it shall be written in the following format:

"%s=\"%s\"\n", <*keyword name*> , <*keyword value*>

If the keyword was `charmap`, the name of the charmap (if any) that was specified via the `localedef` −f option when the locale was created shall be written, with the word `charmap` as <*keyword name*> .

If a value is numeric, it shall be written in one of the following formats:

"%s=%d\n", <*keyword name*> , <*keyword value*>

"%s=%c%o\n", <*keyword name*> , <*escape character*> , <*keyword value*>

"%s=%cx%x\n", <*keyword name*> , <*escape character*> , <*keyword value*>

where the <*escape character*> is that identified by the `escape_char` keyword in the current locale; see 2.5.2.

Compound keyword values (list entries) shall be separated in the output by semicolons. When included in keyword values, the semicolon, the double-quote, the backslash, and any control character shall be preceded (escaped) with the escape character.

(4) If the −k option is not specified, selected keyword values shall be written, each in the following format:

"%s\n", <*keyword value*>

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

7726 If the keyword was `charmap`, the name of the charmap (if any) that was
7727 specified via the `localedef -f` option when the locale was created shall
7728 be written.

7729 (5) If the `-m` option is specified, then a list of all available charmaps shall be
7730 written, each in the format

7731 `"%s\n"`, <*charmap*>

7732 where <*charmap*> is in a format suitable for use as the option-argument
7733 to the `localedef -f` option.

7734 #### 4.34.6.2 Standard Error

7735 Used only for diagnostic messages.

7736 #### 4.34.6.3 Output Files

7737 None.

7738 ### 4.34.7 Extended Description

7739 None.

7740 ### 4.34.8 Exit Status

7741 The `locale` utility shall exit with one of the following values:

7742 0 All the requested information was found and output successfully.

7743 >0 An error occurred.

7744 ### 4.34.9 Consequences of Errors

7745 Default.

7746 ### 4.34.10 Rationale. *(This subclause is not a part of P1003.2)*

7747 **Examples, Usage**

7748 In the following examples, the assumption is that locale environment variables
7749 are set as follows:

```
7750 LANG=locale_x
7751 LC_COLLATE=locale_y
```

7752 The command:

```
7753 locale
```

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

would result in the following output:

```
LANG=locale_x
LC_CTYPE="locale_x"
LC_COLLATE=locale_y
LC_TIME="locale_x"
LC_NUMERIC="locale_x"
LC_MONETARY="locale_x"
LC_MESSAGES="locale_x"
LC_ALL=
```

The order of presentation of the categories is not specified by this standard.

The command

```
LC_ALL=POSIX locale -ck decimal_point
```

would produce:

```
LC_NUMERIC
decimal_point="."
```

The following command shows an application of `locale` to determine whether a user supplied response is affirmative:

```
if printf "%s\n" "$response" | grep -Eq "$(locale yesexpr)"
then
        affirmative processing goes here
else
        nonaffirmative processing goes here
fi
```

If the **LANG** environment variable is not set or set to an empty value, or one of the **LC_*** environment variables is set to an unrecognized value, the actual locales assumed (if any) are implementation defined as described in 2.6.

Implementations are not required to write out the actual values for keywords in the categories LC_CTYPE and LC_COLLATE; however, they must write out the categories (allowing an application to determine, e.g., which character classes are available).

History of Decisions Made

This command was added in Draft 9 to resolve objections to the lack of a way for applications to determine what locales are available, a way to examine the contents of existing public locales, a way to retrieve specific locale items, and a way to recognize affirmative and negative responses in an international environment.

In Draft 10 it was cut back considerably in answer to balloting objections about its complexity and requirement of features not useful for application programs. The format for the no-arguments case was expanded to show the implied values of the categories as an aid to the novice user; the output was of little more value than that from `env`.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Based on the questionable value in a shell script of getting an entire array of characters back, and the problem of returning a collation description that makes sense, short of a complete `localedef` source, the output from requests for categories LC_CTYPE and LC_COLLATE has been made implementation defined.

The `-m` option has been added to allow applications to query for the existence of charmaps. The output is a list of the charmaps (implementation-supplied and user-supplied, if any) on the system.

The `-c` option was included for readability when more than one category is selected (e.g., via more than one keyword name or via a category name). It is valid both with and without the `-k` option.

The `charmap` keyword, which returns the name of the charmap (if any) that was used when the current locale was created, was introduced to allow applications needing the information to retrieve it.

4.35 `localedef` — Define locale environment

4.35.1 Synopsis

`localedef [-c][-f` *charmap* `][-i` *sourcefile* `]` *name*

4.35.2 Description

The `localedef` utility shall convert source definitions for locale categories into a format usable by the functions and utilities whose operational behavior is determined by the setting of the locale environment variables defined in 2.5. It is implementation defined whether users shall have the capability to create new locales, in addition to those supplied by the implementation. If the symbolic constant {POSIX 2_LOCALEDEF } is defined, then the system supports the creation of new locales. In a system not supporting this capability, the `localedef` utility shall terminate with an exit code of 3.

The utility shall read source definitions for one or more locale categories belonging to the same locale from the file named in the `-i` option (if specified) or from standard input.

The *name* operand identifies the target locale. The utility shall support the creation of *public* , or generally accessible locales, as well as *private* , or restricted-access locales. Implementations may restrict the capability to create or modify public locales to users with the appropriate privileges.

Each category source definition shall be identified by the corresponding environment variable name and terminated by an `END` *category-name* statement. The following categories shall be supported. In addition, the input may contain source for implementation-defined categories.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.