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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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]

4.21 env — Set environment for command invocation

4.21.1 Synopsis

env [−i] [*name*=*value*] ... [*utility* [*argument* ...]]

Obsolescent Version :

env [−] [*name*=*value*] ... [*utility* [*argument* ...]]

4.21.2 Description

The env utility shall obtain the current environment, modify it according to its arguments, then invoke the utility named by the *utility* operand with the modified environment.

Optional arguments shall be passed to *utility* .

If no *utility* operand is specified, the resulting environment shall be written to the standard output, with one *name* =*value* pair per line.

4.21.3 Options

The env utility shall conform to the utility argument syntax guidelines described in 2.10.2, except for its nonstandard usage of −, which is obsolescent.

The following options shall be supported by the implementation:

−i Invoke *utility* with exactly the environment specified by the arguments; the inherited environment shall be ignored completely.

− (Obsolescent.) Equivalent to the −i option.

4.21.4 Operands

The following operands shall be supported by the implementation:

name=*value* Arguments of the form *name*=*value* modify the execution environment, and are placed into the inherited environment before the *utility* is invoked.

utility The name of the utility to be invoked. If the *utility* operand names any of the special built-in utilities in 3.14, the results are undefined.

argument A string to pass as an argument for the invoked utility.

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

5614 ### 4.21.5 External Influences

5615 #### 4.21.5.1 Standard Input

5616 None.

5617 #### 4.21.5.2 Input Files

5618 None.

5619 #### 4.21.5.3 Environment Variables

5620 The following environment variables shall affect the execution of `env`:

5621 **LANG** This variable shall determine the locale to use for the
5622 locale categories when both **LC_ALL** and the correspond-
5623 ing environment variable (beginning with **LC_**) do not
5624 specify a locale. See 2.6.

5625 **LC_ALL** This variable shall determine the locale to be used to over-
5626 ride any values for locale categories specified by the set-
5627 tings of **LANG** or any environment variables beginning
5628 with **LC_**.

5629 **LC_CTYPE** This variable shall determine the locale for the interpreta-
5630 tion of sequences of bytes of text data as characters (e.g.,
5631 single- versus multibyte characters in arguments).

5632 **LC_MESSAGES** This variable shall determine the language in which mes-
5633 sages should be written.

5634 **PATH** This variable shall determine the location of the *utility* , as
5635 described in 2.6. If **PATH** is specified as a *name* =*value*
5636 operand to `env`, the *value* given shall be used in the
5637 search for *utility* .

5638 #### 4.21.5.4 Asynchronous Events

5639 Default.

5640 ### 4.21.6 External Effects

5641 #### 4.21.6.1 Standard Output

5642 If no *utility* operand is specified, each *name* =*value* pair in the resulting environ-
5643 ment shall be written in the form:

5644 `"%s=%s\n"`, <*name*> , <*value*>

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

If the *utility* operand is specified, the `env` utility shall not write to standard output.

4.21.6.2 Standard Error

Used only for diagnostic messages.

4.21.6.3 Output Files

None.

4.21.7 Extended Description

None.

4.21.8 Exit Status

If the *utility* utility is invoked, the exit status of `env` shall be the exit status of *utility;* otherwise, the `env` utility shall exit with one of the following values:

0	The `env` utility completed successfully.
1–125	An error occurred in the `env` utility.
126	The utility specified by *utility* was found but could not be invoked.
127	The utility specified by *utility* could not be found.

4.21.9 Consequences of Errors

Default.

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

Examples, Usage

The following command:

```
env -i PATH=/mybin mygrep xyz myfile
```

invokes the command `mygrep` with a new **PATH** value as the only entry in its environment. In this case, **PATH** is used to locate `mygrep`, which then must reside in `/mybin`.

As with all other utilities that invoke other utilities, the standard only specifies what `env` does with standard input, standard output, standard error, input files, and output files. If a utility is executed, it is not constrained by `env`'s specification of input and output.

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

The `command`, `env`, `nohup`, and `xargs` utilities have been specified to use exit code 127 if an error occurs so that applications can distinguish ''failure to find a utility'' from ''invoked utility exited with an error indication.'' The value 127 was chosen because it is not commonly used for other meanings; most utilities use small values for ''normal error conditions'' and the values above 128 can be confused with termination due to receipt of a signal. The value 126 was chosen in a similar manner to indicate that the utility could be found, but not invoked. Some scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for any other reason.

History of Decisions Made

The `-i` option was added to allow the functionality of the – option in a manner compatible with the Utility Syntax Guidelines. It is the intent of the working group that portable applications use the `-i` option, and that in the future the – option be removed from the standard.

Historical implementations of the `env` utility use *execvp* () or *execlp* () (see POSIX .1 {8} 3.1.2) to invoke the specified utility; this provides better performance and keeps users from having to escape characters with special meaning to the shell. Therefore, shell functions, special built-ins, and built-ins that are only provided by the shell are not found. Implementations are free to invoke a shell instead of using one of the *exec* family of routines, but if they do, they must be sure to escape any characters with special meaning to the shell so that the user does not have to be aware of the difference.

Some have suggested that `env` is redundant since the same effect is achieved by:

```
name=value ... utility [argument ...]
```

The example is equivalent to `env` when an environment variable is being added to the environment of the command, but not when the environment is being set to the given value. The `env` utility also writes out the current environment if invoked without arguments. There is sufficient functionality beyond what the example provides to justify inclusion of `env`.

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

4.22 `expr` — Evaluate arguments as an expression

4.22.1 Synopsis

`expr` *operand* ...

4.22.2 Description

The `expr` utility shall evaluate an expression and write the result to standard output.

4.22.3 Options

None.

4.22.4 Operands

The single expression evaluated by `expr` shall be formed from the operands, as described in 4.22.7. Each of the expression operator symbols:

( ) | & = > >= < <= != + − * / % :

and the symbols *integer* and *string* in the table shall be provided by the application as separate arguments to `expr`.

4.22.5 External Influences

4.22.5.1 Standard Input

None.

4.22.5.2 Input Files

None.

4.22.5.3 Environment Variables

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

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

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