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 'env.1p' command

$ man env.1p

ENV(1P) POSIX Programmer's Manual ENV(1P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

env ? set the environment for command invocation

SYNOPSIS

env [-i] [name=value]... [utility [argument...]]

DESCRIPTION

The env utility shall obtain the current environment, modify it accord? ing to its arguments, then invoke the utility named by the utility op? erand 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.

If the first argument is '-', the results are unspecified.

OPTIONS

The env utility shall conform to the Base Definitions volume of POSIX.1?2017, Section 12.2, Utility Syntax Guidelines, except for the unspecified usage of '-'.

The following options shall be supported:

-i Invoke utility with exactly the environment specified by the arguments; the inherited environment shall be ignored com? pletely.

OPERANDS

The following operands shall be supported:

name=value

Arguments of the form name=value shall modify the execution environment, and shall be placed into the inherited environ? ment 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 Section 2.14, Special Built-In Utilities, the results are undefined.

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

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of env:

LANG Provide a default value for the internationalization vari? ables that are unset or null. (See the Base Definitions vol? ume of POSIX.1?2017, Section 8.2, Internationalization Vari? ables for the precedence of internationalization variables used to determine the values of locale categories.)

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).

LC_MESSAGES

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard er? ror.

NLSPATH Determine the location of message catalogs for the processing of LC_MESSAGES.

PATH Determine the location of the utility, as described in the Base Definitions volume of POSIX.1?2017, Chapter 8, Environment Variables. If PATH is specified as a name=value operand to env, the value given shall be used in the search for utility.

ASYNCHRONOUS EVENTS

Default.

STDOUT

If no utility operand is specified, each name=value pair in the resulting environment shall be written in the form:

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

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

STDERR

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

If 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.

CONSEQUENCES OF ERRORS

Default.

The following sections are informative.

APPLICATION USAGE

The command, env, nice, nohup, time, 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 val? ues 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 differ? entiating 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.

Historical implementations of the env utility use the execvp() or exe? clp() functions defined in the System Interfaces volume of POSIX.1?2017 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.

EXAMPLES

The following command:

```
env -i PATH=/mybin:"$PATH" $(getconf V7_ENV) mygrep xyz myfile
```

invokes the command mygrep with a new PATH value as the only entry in its environment other than any variables required by the implementation for conformance. In this case, PATH is used to locate mygrep, which is expected to reside in /mybin.

RATIONALE

As with all other utilities that invoke other utilities, this volume of POSIX.1?2017 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 the specification of input and out? put by env.

The -i option was added to allow the functionality of the removed - op? tion in a manner compatible with the Utility Syntax Guidelines. It is possible to create a non-conforming environment using the -i option, as it may remove environment variables required by the implementation for conformance. The following will preserve these environment variables as well as preserve the PATH for conforming utilities:

```
IFS='
'
# The preceding value should be <space><tab><newline>.
# Set IFS to its default value.
set -f
# disable pathname expansion
\unalias -a
# Unset all possible aliases.
# Note that unalias is escaped to prevent an alias
# being used for unalias.
# This step is not strictly necessary, since aliases are not inherited,
# and the ENV environment variable is only used by interactive shells,
# the only way any aliases can exist in a script is if it defines them
# itself.
unset -f env getconf
# Ensure env and getconf are not user functions.
env -i $(getconf V7_ENV) PATH="$(getconf PATH)" command
```

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.

None.

SEE ALSO

Section 2.14, Special Built-In Utilities, Section 2.5, Parameters and Variables

The Base Definitions volume of POSIX.1?2017, Chapter 8, Environment Variables, Section 12.2, Utility Syntax Guidelines