IEEE P1003.2 Draft 11.2 – September 1991

4.19 echo — Write arguments to standard output

4.19.1 Synopsis

`echo [`*string* ...`]`

4.19.2 Description

The `echo` utility shall write its arguments to standard output, followed by a `<newline>` character. If there are no arguments, only the `<newline>` character shall be written.

4.19.3 Options

The `echo` utility shall not recognize the `--` argument in the manner specified by utility syntax guideline 10 in 2.10.2; `--` shall be recognized as a string operand.

Implementations need not support any options.

4.19.4 Operands

The following operands shall be supported by the implementation:

string A string to be written to standard output. If the first operand is `"-n"` or if any of the operands contain a backslash (\) character, the results are implementation defined.

4.19.5 External Influences

4.19.5.1 Standard Input

None.

4.19.5.2 Input Files

None.

4.19.5.3 Environment Variables

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

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.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_MESSAGES This variable shall determine the language in which diagnostic messages should be written.

4.19.5.4 Asynchronous Events

Default.

4.19.6 External Effects

4.19.6.1 Standard Output

The `echo` utility arguments shall be separated by single `<space>`s and a `<newline>` character shall follow the last argument.

4.19.6.2 Standard Error

Used only for diagnostic messages.

4.19.6.3 Output Files

None.

4.19.7 Extended Description

None.

4.19.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

4.19.9 Consequences of Errors

Default.

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

Examples, Usage

As specified by this standard, echo writes its arguments in the simplest of ways. The two different historical versions of echo vary in fatal incompatible ways.

The BSD echo checks the first argument for the string "−n", which causes it to suppress the <newline> character that would otherwise follow the final argument in the output.

The System V echo does not support any options, but allows escape sequences within its operands:

\a Write an <alert> character.

\b Write a <backspace> character.

\c Suppress the <newline> character that otherwise follows the final argument in the output. All characters following the \c in the arguments are ignored.

\f Write a <form-feed> character.

\n Write a <newline> character.

\r Write a <carriage-return> character.

\t Write a <tab> character.

\v Write a <vertical-tab> character.

\\ Write a backslash character.

\0*num*
Write an 8-bit value that is the 1-, 2-, or 3-digit octal number *num* .

It is not possible to use echo portably across these two implementations unless both −n (as the first argument) and escape sequences are omitted.

The printf utility (see 4.50) can be used to portably emulate any of the traditional behaviors of the echo utility as follows:

— The System V echo is equivalent to:

```
printf "%b\n" "$*"
```

— The BSD echo is equivalent to:

```
if [ "X$1" = "X-n" ]
then
        shift
        printf "%s" "$*"
else
        printf "%s\n" "$*"
fi
```

The echo utility does not support utility syntax guideline 10 because existing applications depend on echo to echo *all* of its arguments, except for the −n option in the BSD version.

New applications are encouraged to use `printf` instead of `echo`. The `echo` utility has not been made obsolescent because of its extremely widespread use in existing applications.

History of Decisions Made

In Draft 8, an attempt was made to merge the extensions of BSD and System V, supporting both `−n` and escape sequences. During initial ballot resolution, a `−e` option was proposed to enable the escape conventions. Both attempts failed, as there are historical scripts that would be broken by any attempt at reconciliation. Therefore, in Draft 9 only the simplest version of `echo` is presented. Implementation-defined extensions on BSD and System V will keep historical applications content. Portable applications that wish to do prompting without `<newline>`s or that could possibly be expecting to echo a `"−n"`, should use the new `printf` utility (see 4.50), derived from the Ninth Edition.

The **LC_CTYPE** variable is not cited because `echo`, as specified here, does not need to understand the characters in its arguments. The System V and BSD implementations might need to be sensitive to it because of their extensions.

4.20 ed — Edit text

4.20.1 Synopsis

`ed [−p` *string* `] [−s] [`*file*`]`

Obsolescent Version :

`ed [−p` *string* `] [−] [`*file*`]`

4.20.2 Description

The `ed` utility is a line-oriented text editor that shall use two modes: *command mode* and *input mode* . In command mode the input characters shall be interpreted as commands, and in input mode they shall be interpreted as text. See 4.20.7.

4.20.3 Options

The `ed` utility shall conform to the utility argument syntax guidelines described in 2.10.2, except for its nonstandard usage of – in the obsolescent version.

The following options shall be supported by the implementation: