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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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

4.19.10 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

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

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:

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

5002 `-p` *string* Use *string* as the prompt string when in command mode. By
5003 default, there shall be no prompt string.

5004 `-s` Suppress the writing of byte counts by `e`, `E`, `r`, and `w` commands
5005 and of the `!` prompt after a `!`*command* .

5006 `-` (Obsolescent.) Equivalent to the `-s` option.

5007 ### 4.20.4 Operands

5008 The following operand shall be supported by the implementation:

5009 *file* If the *file* argument is given, `ed` shall simulate an `e` command on
5010 the file named by the pathname, *file* , before accepting commands
5011 from the standard input.

5012 ### 4.20.5 External Influences

5013 #### 4.20.5.1 Standard Input

5014 The standard input shall be a text file consisting of commands, as described in
5015 4.20.7.

5016 #### 4.20.5.2 Input Files

5017 The input files shall be text files.

5018 #### 4.20.5.3 Environment Variables

5019 The following environment variables shall affect the execution of `ed`:

5020 **HOME** This variable shall determine the pathname of the user's
5021 home directory.

5022 **LANG** This variable shall determine the locale to use for the
5023 locale categories when both **LC_ALL** and the correspond-
5024 ing environment variable (beginning with **LC_**) do not
5025 specify a locale. See 2.6.

5026 **LC_ALL** This variable shall determine the locale to be used to over-
5027 ride any values for locale categories specified by the set-
5028 tings of **LANG** or any environment variables beginning
5029 with **LC_**.

5030 **LC_COLLATE** This variable shall determine the locale for the behavior of
5031 ranges, equivalence classes, and multicharacter collating
5032 elements within regular expressions.

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

LC_CTYPE This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and input files), the behavior of character classes within regular expressions.

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

4.20.5.4 Asynchronous Events

The `ed` utility shall take the standard action for all signals (see 2.11.5.4), with the following exceptions:

SIGINT The `ed` utility shall interrupt its current activity, write the string

```
"?\n"
```

to standard output, and return to command mode (see 4.20.7).

SIGHUP If the buffer is not empty and has changed since the last write, the `ed` utility shall attempt to write a copy of the buffer in a file. First, the file named `ed.hup` in the current directory shall be used; if that fails, the file named `ed.hup` in the directory named by the **HOME** environment variable shall be used. In any case, the `ed` utility shall exit without returning to command mode.

4.20.6 External Effects

4.20.6.1 Standard Output

Various editing commands and the prompting feature (see `-p`) write to standard output, as described in 4.20.7.

4.20.6.2 Standard Error

Used only for diagnostic messages.

4.20.6.3 Output Files

The output files shall be text files whose formats are dependent on the editing commands given.

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

4.20.7 Extended Description

The ed utility shall operate on a copy of the file it is editing; changes made to the copy shall have no effect on the file until a w (write) command is given. The copy of the text is called the *buffer* in this clause, although no attempt is made to imply a specific implementation.

Commands to ed have a simple and regular structure: zero, one, or two *addresses* followed by a single-character *command* , possibly followed by parameters to that command. These addresses specify one or more lines in the buffer. Every command that requires addresses has default addresses, so that the addresses very often can be omitted. If the –p option is specified, the prompt string shall be written to standard output before each command is read.

In general, only one command can appear on a line. Certain commands allow text to be input. This text is placed in the appropriate place in the buffer. While ed is accepting text, it is said to be in *input mode* . In this mode, no commands shall be recognized; all input is merely collected. Input mode is terminated by entering a line consisting of two characters: a period (.) followed by a <newline>. This line is not considered part of the input text.

4.20.7.1 ed Regular Expressions

The ed utility shall support basic regular expressions, as described in 2.8.3. Since regular expressions in ed are always matched against single lines, never against any larger section of text, there is no way for a regular expression to match a <newline>. A null RE shall be equivalent to the last RE encountered.

Regular expressions are used in addresses to specify lines, and in some commands (for example, the s substitute command) to specify portions of a line to be substituted.

4.20.7.2 ed Addresses

Addressing in ed relates to the *current line* . Generally, the current line is the last line affected by a command. The *current line number* is the address (line number) of the current line. The exact effect on the current line number is discussed under the description of each command. The f, h, H, k, P, w, =, and ! commands shall not modify the current line number.

Addresses are constructed as follows:

(1) The character . (period) shall address the current line.

(2) The character $ shall address the last line of the buffer.

(3) A positive decimal number *n* shall address the *n*-th line of the buffer. The first line in the buffer is line number 1.

(4) 'x shall address the line marked with the mark name character *x,* which shall be a lowercase letter from the portable character set. Lines can be marked with the k command described in 4.20.7.3.13.

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

(5) An RE enclosed by slashes (/) shall address the first line found by searching forward from the line following the current line toward the end of the buffer and stopping at the first line containing a string matching the RE. [As stated in 4.20.7.1, an address consisting of a null RE delimited by slashes (//) shall address the next line containing the last RE encountered.] If necessary, the search shall wrap around to the beginning of the buffer and continue up to and including the current line, so that the entire buffer is searched. Within the RE, the sequence \/ shall represent a literal slash instead of the RE delimiter.

(6) An RE enclosed in question-marks (?) shall address the first line found by searching backward from the line preceding the current line toward the beginning of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the end of the buffer and continues up to and including the current line. Within the RE, the sequence \? shall represent a literal question-mark instead of the RE delimiter.

(7) An address followed by a plus sign (+) or a minus sign (-) followed by a decimal number specifies that address plus (respectively minus) the indicated number of lines. The plus sign can be omitted.

(8) If an address begins with + or -, the addition or subtraction is taken with respect to the current line number; for example, –5 is understood to mean .–5.

(9) If an address ends with + or -, then 1 shall be added to or subtracted from the address, respectively. As a consequence of this rule and of rule (8) immediately above, the address – shall refer to the line preceding the current line. Moreover, trailing + and – characters shall have a cumulative effect, so –– shall refer to the current line number less 2.

(10) A comma (,) shall stand for the address pair 1,$, while a semicolon (;) shall stand for the pair .,$.

Commands require zero, one, or two addresses. Commands that require no addresses shall regard the presence of an address as an error. Commands that accept one or two addresses assume default addresses when no addresses are given, as described in 4.20.7.3. If one address is given to a command that allows two addresses, the command shall operate as if it were specified as:

given_address ;. *command*

If more addresses are given than such a command requires, the results are undefined.

Typically, addresses are separated from each other by a comma. They can also be separated by a semicolon. In the latter case, the current line number (.) shall be set to the first address, and only then shall the second address be calculated. This feature can be used to determine the starting line for forward and backward searches [see rules (5) and (6) above]. The second address of any two-address sequence shall correspond to a line that does not precede, in the buffer, the line corresponding to the first address.

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

4.20.7.3 ed Commands

In the following list of ed commands, the default addresses are shown in parentheses. The number of addresses shown in the default shall be the number expected by the command. The parentheses are not part of the address; they show that the given addresses are the default.

It is generally invalid for more than one command to appear on a line. However, any command (except e, E, f, q, Q, r, w, and !) can be suffixed by the letter l, n, or p; in which case, except for the l, n, and p commands, the command shall be executed and then the new current line shall be written as described below under the l, n, and p commands. When an l, n, or p suffix is used with an l, n, or p command, the command shall write to standard output as described below, but it is unspecified whether the suffix writes the current line again in the requested format or whether the suffix has no effect. For example, the pl command (base p command with an l suffix) shall either write just the current line or shall write it twice—once as specified for p and once as specified for l. Also, the g, G, v, and V commands shall take a command as a parameter.

Each address component can be preceded by zero or more <blank>s. The command letter can be preceded by zero or more <blank>s. If a suffix letter (l, n, or p) is given, it shall immediately follow the command.

The e, E, f, r, and w commands shall take an optional *file* parameter, separated from the command letter by one or more <blank>s.

If changes have been made in the buffer since the last w command that wrote the entire buffer, ed shall warn the user if an attempt is made to destroy the editor buffer via the e or q commands. The ed utility shall write the string:

```
"?\n"
```

(followed by an explanatory message if *help mode* has been enabled via the H command) to standard output and shall continue in command mode with the current line number unchanged. If the e or q command is repeated with no intervening command, it shall take effect.

If an end-of-file is detected on standard input when a command is expected, the ed utility shall act as if a q command had been entered.

If the closing delimiter of an RE or of a replacement string (e.g., /) in a g, G, s, v, or V command would be the last character before a <newline>, that delimiter can be omitted, in which case the addressed line shall be written. For example, the following pairs of commands are equivalent:

```
s/s1/s2    s/s1/s2/p
g/s1       g/s1/p
?s1        ?s1?
```

If an invalid command is entered, ed shall write the string:

```
"?\n"
```

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

(followed by an explanatory message if *help mode* has been enabled via the `H` command) to standard output and shall continue in command mode with the current line number unchanged.

4.20.7.3.1 Append Command

Synopsis : `(.)a`
<text>
`.`

The *append* command shall read the given text and append it after the addressed line; the current line number shall become the address of the last inserted line, or, if there were none, the addressed line. Address 0 shall be valid for this command: it shall cause the ‘‘appended’’ text to be placed at the beginning of the buffer.

4.20.7.3.2 Change Command

Synopsis : `(.,.)c` 1
<text>
`.`

The *change* command shall delete the addressed lines, then accept input text that replaces these lines; the current line shall be set to the address of the last line input; or, if there were none, at the line after the last line deleted; if the lines deleted were originally at the end of the buffer, the current line number shall be set to the address of the new last line; if no lines remain in the buffer, the current line number shall be set to zero.

4.20.7.3.3 Delete Command

Synopsis : `(.,.)d`

The *delete* command shall delete the addressed lines from the buffer. The address of the line after the last line deleted shall become the current line number; if the lines deleted were originally at the end of the buffer, the current line number shall be set to the address of the new last line; if no lines remain in the buffer, the current line number shall be set to zero.

4.20.7.3.4 Edit Command

Synopsis : `e` [*file*]

The *edit* command shall delete the entire contents of the buffer and then read in the file named by the pathname *file*. The current line number shall be set to the address of the last line of the buffer. If no pathname is given, the currently remembered pathname, if any, shall be used (see the `f` command). The number of bytes read shall be written to standard output, unless the `-s` option was specified, in the following format:

`"%d\n"`, *<number of bytes read>*

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

The name *file* shall be remembered for possible use as a default pathname in subsequent `e`, `E`, `r`, and `w` commands. If *file* is replaced by `!`, the rest of the line shall be taken to be a shell command line whose output is to be read. Such a shell command line shall not be remembered as the current *file*. All marks shall be discarded upon the completion of a successful `e` command. If the buffer has changed since the last time the entire buffer was written, the user shall be warned, as described previously.

4.20.7.3.5 Edit Without Checking Command

Synopsis : `E` [*file*]

The *Edit* command shall possess all properties and restrictions of the `e` command except that the editor shall not check to see if any changes have been made to the buffer since the last `w` command.

4.20.7.3.6 File-Name Command

Synopsis : `f` [*file*]

If *file* is given, the file-name command shall change the currently remembered pathname to *file*; whether the name is changed or not, it then shall write the (possibly new) currently remembered pathname to the standard output in the following format:

`"%s\n"`, <*pathname*>

The current line number shall be unchanged.

4.20.7.3.7 Global Command

Synopsis : `(1,$)g/` *RE/command list*

In the *global* command, the first step shall be to mark every line that matches the given *RE*. Then, for every such line, the given *command list* shall be executed with the current line number set to the address of that line. When the `g` command completes, the current line number shall have the value assigned by the last command in the command list. If there were no matching lines, the current line number shall not be changed. A single command or the first of a list of commands shall appear on the same line as the global command. All lines of a multiline list except the last line shall be ended with a backslash; the `a`, `i`, and `c` commands and associated input are permitted. The `.` terminating input mode can be omitted if it would be the last line of the *command list*. An empty *command list* shall be equivalent to the `p` command. The use of the `g`, `G`, `v`, `V`, and `!` commands in the *command list* produces undefined results. Any character other than `<space>` or `<newline>` can be used instead of a slash to delimit the *RE*. Within the RE, the RE delimiter itself can be used as a literal character if it is preceded by a backslash.

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

4.20.7.3.8 Interactive Global Command

Synopsis : `(1,$)G/` *RE*`/`

In the *interactive global* command, the first step shall be to mark every line that matches the given *RE*. Then, for every such line, that line shall be written, the current line number shall be set to the address of that line, and any one command (other than one of the `a`, `c`, `i`, `g`, `G`, `v`, and `V` commands) can be input and shall be executed. A `<newline>` shall act as a null command (causing no action to be taken on the current line); an `&` shall cause the reexecution of the most recent nonnull command executed within the current invocation of `G`. Note that the commands input as part of the execution of the `G` command can address and affect any lines in the buffer. The final value of the current line number shall be the value set by the last command successfully executed. (Note that the last command successfully executed shall be the `G` command itself if a command fails or the null command is specified.) If there were no matching lines, the current line number shall not be changed. The `G` command can be terminated by a SIGINT signal. Any character other than `<space>` or `<newline>` can be used instead of a slash to delimit the *RE* and the replacement. Within the RE, the RE delimiter itself can be used as a literal character if it is preceded by a backslash.

4.20.7.3.9 Help Command

Synopsis : `h`

The *help* command shall write a short message to standard output that explains the reason for the most recent `?` notification. The current line number shall be unchanged.

4.20.7.3.10 Help-Mode Command

Synopsis : `H`

The *Help* command shall cause `ed` to enter a mode in which help messages (see the `h` command) shall be written to standard output for all subsequent `?` notifications. The `H` command alternately shall turn this mode on and off; it shall be initially off. If the help-mode is being turned on, the `H` command also shall explain the previous `?` notification, if there was one. The current line number shall be unchanged.

4.20.7.3.11 Insert Command

Synopsis : `(.)i`
<text>
`.`

The *insert* command shall insert the given text before the addressed line; `.` shall be left at the last inserted line, or, if there was none, at the addressed line. This command differs from the `a` command only in the placement of the input text. Address 0 shall be invalid for this command.

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

5298 #### 4.20.7.3.12 Join Command

5299 *Synopsis* : `(.,.+1)j`

5300 The *join* command shall join contiguous lines by removing the appropriate `<new-`
5301 `line>` characters. If exactly one address is given, this command shall do nothing.
5302 If lines are joined, the current line number shall be set to the address of the
5303 joined line; otherwise, the current line number shall be unchanged.

5304 #### 4.20.7.3.13 Mark Command

5305 *Synopsis* : `(.)k`*x*

5306 The *mark* command shall mark the addressed line with name *x*, which shall be a
5307 lowercase letter from the portable character set. The address `'`*x* then shall refer
5308 to this line; the current line number shall be unchanged.

5309 #### 4.20.7.3.14 List Command

5310 *Synopsis* : `(.,.)l`

5311 The *list* command shall write to standard output the addressed lines in a visually 1
5312 unambiguous form. The characters listed in Table 2-15 (see 2.12) shall be written 1
5313 as the corresponding escape sequence. Nonprintable characters not in Table 2-15 1
5314 shall be written as one three-digit octal number (with a preceding `<backslash>`) 1
5315 for each byte in the character (most significant byte first). If the size of a byte on 1
5316 the system is greater than nine bits, the format used for nonprintable characters 1
5317 is implementation defined. 1

5318 Long lines shall be folded, with the point of folding indicated by writing 1
5319 `<backslash><newline>`; the length at which folding occurs is unspecified, but 1
5320 should be appropriate for the output device. The end of each line shall be marked 1
5321 with a `$`. An `l` command can be appended to any other command other than `e`, `E`, 1
5322 `f`, `q`, `Q`, `r`, `w`, or `!`. The current line number shall be set to the address of the last
5323 line written.

5324 #### 4.20.7.3.15 Move Command

5325 *Synopsis* : `(.,.)m`*address*

5326 The *move* command shall reposition the addressed line(s) after the line addressed
5327 by *address* . Address 0 shall be valid for *address* and cause the addressed line(s) to
5328 be moved to the beginning of the buffer. It shall be an error if address *address*
5329 falls within the range of moved lines. The current line number shall be set to the
5330 address of the last line moved.

5331 #### 4.20.7.3.16 Number Command

5332 *Synopsis* : `(.,.)n`

5333 The *number* command shall write to standard output the addressed lines, preced-
5334 ing each line by its line number and a `<tab>` character; the current line number
5335 shall be set to the address of the last line written. The `n` command can be

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

appended to any other command other than `e`, `E`, `f`, `q`, `Q`, `r`, `w`, or `!`.

4.20.7.3.17 Print Command

Synopsis : `(.,.)p`

The *print* command shall write to standard output the addressed lines; the current line number shall be set to the address of the last line written. The `p` command can be appended to any other command other than `e`, `E`, `f`, `q`, `Q`, `r`, `w`, or `!`.

4.20.7.3.18 Prompt Command

Synopsis : `P`

The *Prompt* command shall cause `ed` to prompt with an asterisk (*) (or *string* , if `-p` is specified) for all subsequent commands. The `P` command alternately shall turn this mode on and off; it shall be initially on if the `-p` option is specified, otherwise off. The current line number shall be unchanged.

4.20.7.3.19 Quit Command

Synopsis : `q`

The *quit* command shall cause `ed` to exit. If the buffer has changed since the last time the entire buffer was written, the user shall be warned, as described previously.

4.20.7.3.20 Quit Without Checking Command

Synopsis : `Q`

The *Quit* command shall cause `ed` to exit without checking if changes have been made in the buffer since the last `w` command.

4.20.7.3.21 Read Command

Synopsis : `($)r` [*file*]

The *read* command shall read in the file named by the pathname *file* and append it after the addressed line. If no *file* argument is given, the currently remembered pathname, if any, shall be used (see `e` and `f` commands). The currently remembered pathname shall not be changed unless there is no remembered pathname. Address 0 shall be valid for `r` and shall cause the file to be read at the beginning of the buffer. If the read is successful, and `-s` was not specified, the number of bytes read shall be written to standard output in the following format:

`"%d\n"`, *<number of bytes read>*

The current line number shall be set to the address of the last line read in. If *file* is replaced by `!`, the rest of the line shall be taken to be a shell command line whose output is to be read. Such a shell command line shall not be remembered as the current pathname.

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

5372 #### 4.20.7.3.22 Substitute Command

5373 *Synopsis* : `(.,.)s/` *RE/replacement/flags*

5374 The *substitute* command shall search each addressed line for an occurrence of the
5375 specified RE and replace either the first or all (nonoverlapped) matched strings
5376 with the *replacement* ; see the following description of the `g` suffix. It is an error if
5377 the substitution fails on every addressed line. Any character other than `<space>`
5378 or `<newline>` can be used instead of a slash to delimit the *RE* and the replace-
5379 ment. Within the RE, the RE delimiter itself can be used as a literal character if it
5380 is preceded by a backslash. The current line shall be set to the address of the last
5381 line on which a substitution occurred.

5382 An ampersand (&) appearing in the *replacement* shall be replaced by the string
5383 matching the RE on the current line. The special meaning of & in this context can
5384 be suppressed by preceding it by backslash. As a more general feature, the char-
5385 acters *n*, where *n* is a digit, shall be replaced by the text matched by the
5386 corresponding backreference expression (see 2.8.3.3). When the character `%` is the
5387 only character in the *replacement* , the *replacement* used in the most recent substi-
5388 tute command shall be used as the *replacement* in the current substitute com-
5389 mand; if there was no previous substitute command, the use of `%` in this manner
5390 shall be an error. The `%` shall lose its special meaning when it is in a replacement
5391 string of more than one character or is preceded by a backslash.

5392 A line can be split by substituting a `<newline>` character into it. The application 1
5393 shall escape the `<newline>` in the *replacement* by preceding it by backslash. 1
5394 Such substitution cannot be done as part of a `g` or `v` command list. The current
5395 line number shall be set to the address of the last line on which a substitution is
5396 performed. If no substitution is performed, the current line number shall be
5397 unchanged. If a line is split, a substitution shall be considered to have been per-
5398 formed on each of the new lines for the purpose of determining the new current
5399 line number. A substitution shall be considered to have been performed even if
5400 the replacement string is identical to the string that it replaces.

5401 The value of *flags* shall be zero or more of:

5402 *count* Substitute for the *count* th occurrence only of the *RE* found on each
5403 addressed line.

5404 `g` Globally substitute for all nonoverlapping instances of the *RE* rather
5405 than just the first one. If both `g` and *count* are specified, the results
5406 are unspecified.

5407 `l` Write to standard output the final line in which a substitution was
5408 made. The line shall be written in the format specified for the `l` com-
5409 mand.

5410 `n` Write to standard output the final line in which a substitution was
5411 made. The line shall be written in the format specified for the `n` com-
5412 mand.

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

`p` Write to standard output the final line in which a substitution was made. The line shall be written in the format specified for the `p` command.

4.20.7.3.23 Copy Command

Synopsis : `(.,.)t`*address*

The `t` command shall be equivalent to the `m` command, except that a copy of the addressed lines shall be placed after address *address* (which can be 0); the current line number shall be set to the address of the last line added.

4.20.7.3.24 Undo Command

Synopsis : `u`

The *undo* command shall nullify the effect of the most recent command that modified anything in the buffer, namely the most recent `a`, `c`, `d`, `g`, `i`, `j`, `m`, `r`, `s`, `t`, `u`, `v`, `G`, or `V` command. All changes made to the buffer by a `g`, `G`, `v`, or `V` global command shall be ‘‘undone’’ as a single change; if no changes were made by the global command (such as with `g/`*RE*`/p`), the `u` command shall have no effect. The current line number shall be set to the value it had immediately before the command being undone started.

4.20.7.3.25 Global Non-Matched Command

Synopsis : `(1,$)v/`*RE/command list*

This command shall be equivalent to the global command `g` except that the lines that are marked during the first step shall be those that do not match the RE.

4.20.7.3.26 Interactive Global Not-Matched Command

Synopsis : `(1,$)V/`*RE/*

This command shall be equivalent to the interactive global command `G` except that the lines that are marked during the first step shall be those that do not match the RE.

4.20.7.3.27 Write Command

Synopsis : `(1,$)w [`*file*`]`

The *write* command shall write the addressed lines into the file named by the pathname *file*. The command shall create the file, if it does not exist, or shall replace the contents of the existing file. The currently remembered pathname shall not be changed unless there is no remembered pathname. If no pathname is given, the currently remembered pathname, if any, shall be used (see `e` and `f` commands); the current line number shall be unchanged. If the command is successful, the number of bytes written shall be written to standard output, unless the `−s` option was specified, in the following format:

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

5449 "%d\n", <*number of bytes written*>

5450 If *file* begins with `!`, the rest of the line shall be taken to be a shell command line
5451 whose standard input shall be the addressed lines. Such a shell command line
5452 shall not be remembered as the current pathname. This usage of the write com- 1
5453 mand with `!` shall not be considered as a ‘‘last `w` command that wrote the entire 1
5454 buffer,’’ as described previously; thus, this alone shall not prevent the warning to 1
5455 the user if an attempt is made to destroy the editor buffer via the `e` or `q` com- 1
5456 mands. 1

5457 #### 4.20.7.3.28 Line Number Command

5458 *Synopsis* : `($)=`

5459 The line number of the addressed line shall be written to standard output in the
5460 following format:

5461 "%d\n", <*line number*>

5462 The current line number shall be unchanged by this command.

5463 #### 4.20.7.3.29 Shell Escape Command

5464 *Synopsis* : `!`*command*

5465 The remainder of the line after the `!` shall be sent to the command interpreter to
5466 be interpreted as a shell command line. Within the text of that shell command
5467 line, the unescaped character `%` shall be replaced with the remembered pathname;
5468 if a `!` appears as the first character of the command, it shall be replaced with the
5469 text of the previous shell command executed via `!`. Thus, `!!` shall repeat the pre-
5470 vious `!`*command* . If any replacements of `%` and/or `!` are performed, the modified 2
5471 line shall be written to the standard output before *command* is executed. The `!` 2
5472 command shall write 2

5473 "!\n"

5474 to standard output upon completion, unless the `-s` option is specified. The
5475 current line number shall be unchanged.

5476 #### 4.20.7.3.30 Null Command

5477 *Synopsis* : `(.+1)`

5478 An address alone on a line shall cause the addressed line to be written. A `<new-`
5479 `line>` alone shall be equivalent to `.+1p`. The current line number shall be set to
5480 the address of the written line.

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

4.20.8 Exit Status

The ed utility shall exit with one of the following values:

0 Successful completion without any file or command errors.

>0 An error occurred.

4.20.9 Consequences of Errors

When an error in the input script is encountered, or when an error is detected that is a consequence of the data (not) present in the file or due to an external condition such as a read or write error:

— If the standard input is a terminal device file, all input shall be flushed, and a new command read.

— If the standard input is a regular file, ed shall terminate with a nonzero exit status.

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

Examples, Usage

Some historical implementations contained a bug that allowed a single period to be entered in input mode as <backslash> <period> <newline>. This is not allowed by the POSIX.2 ed because there is no description of escaping any of the characters in input mode; backslashes are entered into the buffer exactly as typed. The typical method of entering a single period has been to precede it with another character and then use the substitute command to delete that character.

Because of the extremely terse nature of the default error messages, the prudent script writer will begin the ed input commands with an H command, so that if any errors do occur at least some clue as to the cause will be made available.

History of Decisions Made

The initial description of this utility was adapted from the *SVID*. It contains some features not found in Version 7 or BSD-derived systems. Some of the differences between the POSIX.2 and BSD ed utilities include, but need not be limited to:

— The BSD - option does not suppress the ! prompt after a ! command.

— BSD does not support the special meanings of the % and ! characters within a ! command.

— BSD does not support the *addresses* ; and ,.

— BSD allows the command/suffix pairs pp, ll, etc., which are unspecified in POSIX.2.

— BSD does not support the ! character part of the e, r, or w commands.

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

— A failed `g` command in BSD sets the line number to the last line searched if there are no matches.

— BSD does not default the command list to the `p` command.

— BSD does not support the `G`, `h`, `H`, `n`, or `V` commands.

— On BSD, if there is no inserted text, the insert command changes the current line to the referenced line −1; i.e., the line before the specified line.

— On BSD, the join command with only a single address changes the current line to that address.

— BSD does not support the `P` command; moreover, in BSD it is synonymous with the `p` command.

— BSD does not support the *undo* of the commands `j`, `m`, `r`, `s`, or `t`.

— The BSD `ed` commands `W`, `wq`, and `z` are not present in POSIX.2.

The `-s` 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 `-s` option, and that in the future the − option be removed from the standard.

Prior to Draft 8 there was a limit, {ED_FILE_MAX}, which described the historical limitations of some `ed` utilities in their handling of large files; some of these have had problems with files in the >100 KB range. It was this limitation that prompted much of the desire to include a `split` command in the standard. Since this limit was removed, the standard requires that implementations document the file size limits imposed by `ed` in the conformance document. The limit {ED_LINE_MAX} was also removed; therefore, the global limit {LINE_MAX} is used for input and output lines.

The `\{`*m*`,`*n*`\}` notation was removed from the description of regular expressions because this functionality is now described in 2.8.3.

The manner in which the `l` command writes nonprintable characters was changed to avoid the historical backspace-overstrike method. On video display terminals, the overstrike is ambiguous because most terminals simply replace overstruck characters, making the `l` format not useful for its intended purpose of unambiguously understanding the content of the line. The historical backslash escapes were also ambiguous. (The string `"a\0011"` could represent a line containing those six characters or a line containing the three characters `'a'`, a byte with a binary value of 1, and a `'1'`.) In the format required here, a backslash appearing in the line will be written as `"\\"` so that the output is truly unambiguous. The method of marking the ends of lines was adopted from the `ex` editor (see the User Portability Extension) and is required for any line ending in <space>s; the `$` is placed on all lines so that a real `$` at the end of a line cannot be misinterpreted.

Systems with bytes too large to fit into three octal digits must devise other means of displaying nonprintable characters. Consideration was given to requiring that the number of octal digits be large enough to hold a byte, but this seemed to be too

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

confusing for applications on the vast majority of systems where three digits are adequate. It would be theoretically possible for the application to use the `getconf` utility to find out the {CHAR_BIT} value and deal with such an algorithm; however, there is really no portable way that an application can use the octal values of the bytes across various coded character sets anyway, so the additional specification did not seem worth the effort.

The description of how a NUL is written was removed. The NUL character cannot be in text files, and the standard should not dictate behavior in the case of undefined, erroneous input.

The text requiring filenames accepted by the `E`, `e`, `R`, and `r` commands to be patterns was removed due to balloting objections that this was undesirable and not existing practice.

The `−p` option in Drafts 8 and 9 said that it only worked when standard input was associated with a terminal device. This has been changed to conform to existing implementations, thereby allowing applications to interpose themselves between a user and the `ed` utility.

The form of the substitute command that uses the *n* suffix was limited to the first 512 matches in a previous draft (where this was described incorrectly as ‘‘backreferencing’’). This limit has been removed because there is no reason an editor processing lines of {LINE_MAX} length should have this restriction. The command `s/x/X/2047` should be able to substitute the 2047th occurrence of `x` on a line.

The use of printing commands with printing suffixes (such as `pn`, `lp`, etc.) was made unspecified because BSD-based systems allow this, whereas System V does not.

Some BSD-based systems exit immediately upon receipt of end-of-file if all of the lines in the file had been deleted. Since POSIX.2 refers to the `q` command in this instance, such behavior is not allowed.

Some historical implementations returned exit status zero even if command errors had occurred; this is not allowed by POSIX.2.

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

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.