IEEE P1003.2a Draft 8 – December 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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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

5.10 ex — Text editor

5.10.1 Synopsis

ex [−rR][−s | −v][−c *command*][−t *tagstring*][−w *size*][*file* ...]

Obsolescent Version :

ex [−rR][− | −v][+*command*][−t *tagstring*][−w *size*][*file* ...] 8

5.10.2 Description

The ex utility is a line-oriented text editor that shall support both line and full-screen editing (see vi in 5.35).

Certain block-mode terminals do not have all the capabilities necessary to support the complete ex definition, such as the full-screen editing commands (*visual* mode) or *open mode* . When these commands cannot be supported on such terminals, this condition shall not produce an error message such as ‘‘not an editor command’’ nor report a syntax error. The implementation may either accept the commands and produce results on the screen that are the result of an unsuccessful attempt to meet the requirements of this standard or report an error describing the terminal-related deficiency. The affected commands are noted as they occur later in this clause. 7 7

5.10.3 Options

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

The following options shall be supported by the implementation:

−c *command*
+*command* (Obsolescent.)
: Begin editing by executing the specified ex command-mode command(s). As with normal editing command-line entries, the *command* option-argument can consist of multiple ex commands separated by vertical-line characters (|). The use of commands that enter input or visual modes in this manner produces undefined results.

−r
: Recover the named files after an editor or system crash, after the editor has been terminated by a signal, or after the use of a preserve editor command. A ‘‘crash’’ in this context is an unexpected failure of the system or utility that requires restarting the failed system or utility. A system crash implies that any utilities running at the time also crash. In the case of an editor or system crash, the degree of recovery (the number of changes to the buffer since the most recent preserve command) available is

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

unspecified.

If no *file* operands are given, all other options, the **EXINIT** variable, and any `.exrc` files shall be ignored; a list of all recoverable files available to the invoking user shall be written; and `ex` shall exit without reading files or processing user commands.

`−R` Set read-only mode, preventing accidental overwriting of the files. Any command that would write to a file shall require the `!` suffix (see, for example, 5.10.7.2.37) to be effective in this mode.

`−s`
`-` (Obsolescent.)
Prepare `ex` for batch use by taking the following actions:

— Suppress writing prompts and informational (but not diagnostic) messages.

— Ignore the value of **TERM** and any implementation default terminal type and assume the terminal is a type incapable of supporting visual mode; see 5.10.7.2.36 and the description of `vi` in 5.35.

— Suppress the use of the **EXINIT** environment variable (see 5.10.5.3) and the reading of any `.exrc` file (see 5.10.7).

`−t` *tagstring*
Edit the file containing the specified *tagstring* and proceed as if the first command were `:tag` *tagstring* . (See `ctags` in 5.7). The tags feature represented by `−t` *tagstring* and the `ta` command (see 5.10.7.2.32) is optional. It shall be provided on any system that also provides a conforming implementation of `ctags`; otherwise, the use of `−t` produces undefined results.

`−v` Invoke `vi` (see 5.35).

`−w` *size* Set the value of the *window* editor option to *size* .

If both the `−t` *tagstring* and `−c` *command* (or the obsolescent `+`*command*) options are given, the `−t` *tagstring* shall be processed first; i.e., the file containing the tag is selected by `−t` and then the command is executed.

5.10.4 Operands

The following operand shall be supported by the implementation:

file A pathname of a file to be edited.

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

5.10.5 External Influences

5.10.5.1 Standard Input

The standard input shall be a text file consisting of commands, as described in 5.10.7.

5.10.5.2 Input Files

Input files shall be text files or files that would be text files except for an incomplete last line that is not longer than {LINE_MAX} − 1 bytes in length and contains no NUL characters. The editing of other forms of files may optionally be allowed by ex implementations.

The .exrc files (see 5.10.7) shall be text files consisting of commands.

By default, ex shall read lines from the files to be edited without interpreting any of those lines as any form of editor command.

5.10.5.3 Environment Variables

The following environment variables shall affect the execution of ex:

COLUMNS This variable shall override the system-selected horizontal screen size. See 2.6 for valid values and results when it is unset or null.

EXINIT This environment variable shall be interpreted to contain a list of ex commands that are executed on editor startup, before reading the first file. The list can contain multiple commands by separating them using a vertical-line (|) character. See 5.10.7 for more details of the initialization phase.

HOME This variable shall be interpreted as a pathname of a directory that shall be searched for an editor startup file named .exrc; see 5.10.7 for details.

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_**.

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

LC_COLLATE This variable shall determine the locale for character collation information in regular expressions.

LC_CTYPE This variable shall determine 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, the classification of characters as upper- or lowercase letters, the case conversion of letters, and the detection of word boundaries.

LC_MESSAGES This variable shall determine the processing of affirmative responses and the language in which messages should be written.

LINES This variable shall override the system-selected vertical screen size, used as the number of lines in a screenful and the vertical screen size in visual mode. See 2.6 for valid values and results when it is unset or null.

PATH This variable shall determine the search path for the shell command specified in the editor commands `shell`, `read`, and `write`; see the description of command search and execution in 3.9.1.1.

SHELL This variable shall be interpreted as the preferred command-line interpreter for use in `!`, `shell`, `read`, and other commands with an operand of the form `!`*string* . For the `shell` command, the program shall be invoked with the single argument `-i`, for all others it shall be invoked with the two arguments `-c` and *string* . If no **SHELL** environment variable is set, or it is set to a null string, the `sh` (see 4.56) utility shall be used.

TERM This variable shall be interpreted as the name of the terminal type. If this variable is unset or null, an unspecified default terminal type shall be used.

5.10.5.4 Asynchronous Events

The following actions shall be taken upon receipt of signals:

SIGINT When an interrupt occurs, `ex` shall alert the terminal and write a message. The current editor command shall be aborted and `ex` shall return to the command level and prompt for another command. If the standard input is not a terminal device, `ex` shall exit at the interrupt and return a nonzero exit status. (The alerting action can be modified by the use of the `errorbells` editor option; see 5.10.7.5.)

SIGCONT The screen shall be refreshed (if in visual mode).

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

SIGHUP If the current buffer has changed since the last `e` or `w` command, `ex` shall attempt to save the current file in a state such that it can be recovered later by an `ex -r` command. 8

The action taken for all other signals is unspecified. 7

5.10.6 External Effects

5.10.6.1 Standard Output

The standard output shall be used only for writing prompts to the user, for informational messages, and for writing lines from the file.

5.10.6.2 Standard Error

Used only for diagnostic messages.

5.10.6.3 Output Files

The output from `ex` shall be text files that are identical to the input files if no changes have been made to the files by commands, with the exception that in all cases where a forced session termination (the `ex` command `q!`; see 5.10.7.2.23) has not been issued prior to any file write, a trailing `<newline>` shall be added to the last line of the file if one was not present in the input.

5.10.7 Extended Description

The pathname of the file being edited by `ex` is referred to as the *current* file. The text of the file shall be read into a working version of the file (called *buffer* in this clause, although no attempt is made to imply a specific implementation) and all editing changes shall be performed on that version; the changes shall have no effect on the original file until an `ex` command causes the file to be written out. Lines in the buffer may each be limited to {LINE_MAX} bytes and an error message may be written if the limit is exceeded during editing.

The *alternate* pathname is the name of the last file mentioned in an editor command, or the previous current pathname if the last file mentioned became the current file. When the character `%` appears in a pathname entered as part of a command argument, it shall be replaced by the current pathname; the character `#` shall be replaced by the alternate pathname. Any character, including `%` and `#`, shall retain its literal value (be escaped) when preceded by a backslash.

When an error occurs, `ex` shall alert the terminal and write a message. (The alerting action can be modified by the use of the `errorbells` editor option; see 5.10.7.5.)

If the system crashes, `ex` shall attempt to preserve the buffer if any unwritten changes were made. The command-line option `-r` can be used to retrieve the saved changes.

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

During initialization, before the first file is read or any user commands from the terminal are processed, if the environment variable **EXINIT** is set, the editor shall execute the `ex` commands contained in that variable. If the variable is not set, `ex` shall attempt to read commands from the file `$HOME/.exrc` (the file `.exrc` in the directory referred to by the **HOME** environment variable). If and only if **EXINIT** or `$HOME/.exrc` sets the editor option `exrc`, `ex` finally shall attempt to read commands from a file `.exrc` in the current directory. In the event that **EXINIT** is not set and the current directory is the user's home directory, any `.exrc` file shall only be processed once. No `.exrc` file shall be read unless it is owned by the same user ID as the effective user ID of the process. After any `.exrc` files are processed, any commands specified by the `-c` option shall be processed.

By default `ex` shall start in the command mode, which shall be indicated by the `":"` prompt. The input mode can be entered by `append`, `insert`, or `change` commands; it can be exited (and command mode re-entered) by typing a period (`.`) alone at the beginning of a line. There is one other mode, visual mode, in which full-screen editing is available. This is described more fully under the `visual` command and in 5.35. The command line can consist of multiple `ex` commands separated by vertical-line characters (`|`). The use of commands that enter input or visual modes in this manner, unless they are the final command on the line, produces undefined results.

Command lines beginning with the double-quote character (`"`) shall be ignored. This can be used for comments in an editor script.

5.10.7.1 Addressing

Addressing in `ex` relates to the *current line*. In general, the current line shall be the last line affected by a command; the exact effect on the current line is discussed under the description of each command. When the buffer contains no lines, the current line shall be set to zero.

Addresses shall be constructed by one of the following methods:

(1) The address `.` (period) refers to the current line.

(2) The address `$` refers to the last line of the buffer.

(3) The address *n*, where *n* shall be a decimal number, refers to the *n*-th line of the buffer.

(4) The address *'x* refers to the line marked with the mark-name character *x*, which shall be a lowercase letter of the POSIX Locale. Lines can be marked with the `ma` or `k` commands described below.

(5) A regular expression (RE) enclosed by slashes (`/`) is an address, and refers to 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. The second slash can be omitted at the end of a command line. If the `wrapscan` option is set, 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. (See

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

5.10.7.3.)

(6) A RE enclosed in question marks (?) addresses 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. The second question mark can be omitted at the end of a command line. If the `wrapscan` option is set, the search shall wrap around from the beginning of the buffer to the end of the buffer and continue up to and including the current line, so that the entire buffer is searched. (See 5.10.7.3.)

(7) An address followed by a plus sign (+) or a minus sign (–) followed by a decimal number is an offset address, and refers to the first address plus (respectively minus) the indicated number of lines. If the address is omitted, the addition or subtraction shall be taken with respect to the current line.

(8) An address of + or – followed by a number shall be taken with respect to the current line number; for example, `–5` is understood to mean `.–5`.

(9) An address ending with + or – shall have 1 added to or subtracted from the address, respectively. As a consequence of this rule and of rule (8) above, the address – refers to the line preceding the current line. Moreover, trailing + and – characters have a cumulative effect; for example, –– refers to the current line less 2.

(10) A percent sign (%) shall stand for the address pair `1,$`.

Commands require zero, one, or two addresses. See the descriptions of *line* and *range* in 5.10.7.2. Commands that require zero addresses shall regard the presence of an address as an error.

Adjacent addresses in a *range* shall be separated from each other by a comma (`,`) or a semicolon (`;`). In the latter case, the current line (`.`) shall be set to the first address, and only then is the second address 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 follows, in the buffer, the line corresponding to the first address. The first address shall be less than or equal to the second address. The first address shall be greater than or equal to the first line of the editing buffer and the last address shall be less than or equal to the last line of the editing buffer. Any other case shall be an error.

5.10.7.2 Command Descriptions

The following symbols are used in this subclause to represent optional modifiers. Any or all can be omitted; the defaults are shown.

line A single line address, given in any of the forms described in 5.10.7.1; the default for *line* shall be the current line.

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

range A *line* , or a pair of line addresses, separated by a comma or semicolon (see 5.10.7.1 for the difference between the two); the default for *range* shall be the current line only (`.,.`). A percent sign (%) shall stand for the range `(1,$)`. If the range specified is such that the starting address exceeds the ending address, the range is invalid and the command shall not be performed. If more than the expected number of addresses are given in a range, the greatest valid number of the last one(s) given shall be used. For example, `1,3,5p` prints lines 3 through 5 (because two is the greatest valid number in the range accepted by `print`).

count A positive integer, specifying the number of lines to be affected by the command; the default for *count* shall be 1.

flags One of the characters #, `p`, or `l` (ell), or both # and `l` to add numbers to list-format output. When a command with such a flag completes, the addressed line(s) shall be written out as if by the corresponding #, `p`, or `l` command. The use of *flags* shall apply to all lines written by the `list`, `number`, `open`, `print`, `substitute`, `visual`, &, and `z` commands; for other commands, it shall apply to the current line at the completion of the command. In addition, any number of + or − characters can also be given after the flags, in which case the line written shall not be the one affected by the command, but rather the line addressed by the offset address as described above. The default for *flags* shall be null.

buffer One of a number of named areas for saving text. The named buffers are specified by the lowercase letters of the POSIX Locale. Specifying *buffer* shall cause the area of text affected by the command to be stored into the buffer as it was before the command took effect. This argument is also used on the `put` command and the visual mode ‘‘put’’ commands (`p` and `P`) to specify the buffer that shall provide the text to insert.

If the buffer name is specified in uppercase, and the buffer is to be modified (as with a deletion or yanking command), the buffer shall be appended to rather than being overwritten. If the buffer is not to be modified (as in a visual mode ‘‘put’’ commands) the buffer name can be specified in lowercase or uppercase with the same results. There shall be also one unnamed buffer, which is the repository for all text deleted (with the `delete` or visual mode `d` command) or yanked (with the `yank` or visual mode `y` command) when no buffer is specified.

There are also numbered buffers, 1 through 9, which shall be accessible only from visual mode. These buffers are special in that, in visual mode, when deleted text is placed in the unnamed buffer, it also shall be placed in buffer 1, the previous contents of buffer 1 shall be placed in buffer 2, and so on. Any text in buffer 9 shall be lost. Text that is yanked (or otherwise copied) into the

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

unnamed buffer shall not modify the numbered buffers. Text cannot be placed directly into the numbered buffers although it can be retrieved from them by using a visual mode ''put'' command with the buffer name given as a number. When the *buffer* modifier is not used in the commands below, the unnamed buffer shall be the default.

file A pattern used to derive a pathname; the default shall be the current file, as defined above. If no current file has yet been established, a warning shall be written and the command shall be aborted, except where specifically noted in the individual command descriptions that follow. The pattern shall be subjected to the process of shell word expansions (see 3.6); if more than a single pathname results and the command is expecting one file, the effects are unspecified.

word In the POSIX Locale, a *word* consists of a maximal sequence of letters, digits, and underscores, delimited at both ends by characters other than letters, digits, or underscores, or by the beginning or end of a line or the file.

! A character that can be appended to the command to modify its operation, as detailed in the individual command descriptions.

If both a *count* and a *range* are specified for a command that uses them, the number of lines affected shall be taken from the *count* value rather than the *range* . The starting line for the command shall be taken to be the first line addressed by the *range* .

When only a *line* or *range* is specified with no command, the implied command shall be either a `print`, `list`, or `number` (`p`, `l`, or #). The command selected shall be the last of these three commands to be used, including use as a *flag* . When no range or count is specified and the command line is a blank line, the current line shall be written, and the current line shall be set to `.+1`.

Zero or more `<blank>` characters can precede or follow the addresses, *count* , *flags* , or command name. Any object following a command name (such as *buffer* , *file* , etc.) that begins with an alphabetic character shall be separated from the command name with at least one `<blank>`.

For each of the commands listed below, the command can be entered as the abbreviation (those characters in the Synopsis command word preceding the `[`), the full command (all characters shown for the command word, omitting the `[` and `]`), or any subset of the characters of the full command down to the abbreviation. For example, the `args` command (shown as `ar[gs]` in the Synopsis) can be entered as `ar`, `arg`, or `args`.

5.10.7.2.1 `abbrev`

Synopsis : `ab[brev]` *word rhs*

Add the named abbreviation to the current abbreviation list. In visual mode, if *word* is typed so that it is preceded and followed by characters that cannot be part

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

of a *word* (as defined previously), it shall be replaced by the string *rhs*.

5.10.7.2.2 append

Synopsis : [*line*] a[ppend][!]

Enter input mode; the input text shall be placed after the specified line. If line 0 is specified, the text shall be placed at the beginning of the buffer. The current line indicator shall be set to the last input line; if no lines are input, it shall be set to the target line, or to the first line of the file if a target of 0 was specified. Following the command name with ! shall cause the `autoindent` editor option setting to be toggled for the duration of this command only.

5.10.7.2.3 args

Synopsis : ar[gs]

Write the argument list with the current argument inside [and]. The argument list is the list of operands on startup, which can subsequently be replaced by the operands of the `next` command.

5.10.7.2.4 change

Synopsis : [*range*] c[hange][!] [*count*]

Enter input mode; the input text shall replace the specified lines (*range*). The current line indicator shall be set to the last line input; if no lines were input, it shall be set to the line before the target line, or to the first line of the file if there are no lines preceding the target. Following the command name with ! shall cause the `autoindent` editor option setting to be toggled for the duration of this command only.

5.10.7.2.5 chdir

Synopsis : chd[ir][!] [*directory*]
Synopsis : cd[!] [*directory*]

Change the current working directory to *directory*. If the current buffer has been modified since the last write, a warning shall be written and the command shall fail. This warning can be overridden by appending ! to the command name, which shall allow the command to complete.

5.10.7.2.6 copy

Synopsis : [*range*] co[py] *line* [*flags*]
Synopsis : [*range*] t *line* [*flags*]

Place a copy of the specified lines (*range*) after the specified destination *line*; line 0 specifies that the lines shall be placed at the beginning of the buffer.

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

5.10.7.2.7 delete

Synopsis : [*range*] d[elete] [*buffer*] [*count*] [*flags*]

Delete the specified lines from the buffer. If a named *buffer* is specified, the deleted text shall be saved in it; otherwise, the deleted text shall be saved in the unnamed buffer. If the command name is followed by a letter that could be interpreted as either a buffer name or a *flag* value (because neither a *count* nor an additional *flags* value was given), ex shall consider the letter to be a *flags* value if the letter directly follows the command name, without any <blank> separation; if the letter is preceded by <blank>(s), it shall be considered a buffer name.

For example:

1dp or 1deletep	Deletes the first line and prints the line that was second.
1d p	Deletes the first line, saving it in buffer p.
1d p1l	(Pee-one-ell.) Deletes the first line, saving it in buffer p, and listing the line that was second.

The current line indicator shall be set to the line following the deleted lines, or to the last line if the deleted lines were at the end.

5.10.7.2.8 edit

Synopsis : e[dit][!] [+*line*] [*file*]
Synopsis : ex[!] [+*line*] [*file*]

Begin editing *file* . If the current buffer has been modified since the last write, a warning shall be written and the command shall be aborted. This action can be overridden by appending the character ! to the command name (e! *file*). The current line indicator shall be the last line of the buffer; however, if this command is executed from within visual mode, the current line indicator shall be the first line of the buffer. If the +*line* option is specified, the current line indicator shall be set to the specified position, where *line* can be a number (or $) or specified as ‘‘/*pattern* ’’ or ‘‘?*pattern* ’’. Preceding the pattern with a / shall start a search from the beginning of the file. Preceding the pattern with a ? shall start a search from the end of the file. This command shall be affected by the editor options autowrite and writeany; see 5.10.7.5 for details.

5.10.7.2.9 file

Synopsis : f[ile] [*file*]

Write the current pathname, the number of lines, the current position when no *file* argument has been specified; ex may write other unspecified information. If no current file has yet been established, an unspecified message shall be written to indicate that no file is being edited. With *file* , ex shall change the current file name to *file* , without changing the contents of the buffer or the previous current file.

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

5.10.7.2.10 `global`

Synopsis : [*range*] g[lobal] /*pattern* / [*cmds*]
Synopsis : [*range*] v /*pattern* / [*cmds*]

Mark the lines within the given range that match (g) or do not match (v) the given pattern. Then execute the ex command(s) given by *cmds* with the current line (.) set to each marked line.

Multiple *cmds* can be specified, one per line, by escaping each <newline> with a backslash. If *cmds* are omitted, each line shall be written. For the append, change, and insert commands, the input text shall be included as part of the global command line; in this case the terminating period can be omitted if it ends *cmds* . The visual command can be specified as one of the *cmds* . In this mode, input shall be taken from the terminal. Entering a Q from visual mode shall cause the next line matching the pattern to be selected and visual mode to be reentered, until the list is exhausted.

The global command itself and the undo command cannot be used in *cmds* . The editor options autoprint, autoindent, and report shall be inhibited for the duration of the g or v command.

5.10.7.2.11 `insert`

Synopsis : [*line*] i[nsert][!]

Enter input mode; the input text shall be placed before the specified line. The current line indicator shall be set to the last line input; if no lines were input, it shall be set to the line before the target line, or to the first line of the file if there are no lines preceding the target. Following the command name with the character ! shall cause the autoindent editor option setting to be toggled for the duration of this command only.

5.10.7.2.12 `join`

Synopsis : [*range*] j[oin][!] [*count*] [*flags*]

Join the text from the specified lines together into one line. In the POSIX Locale, when the last character on the first line of a pair of lines to be joined is a period, two <space>s shall be added following the period; when the last character of the first line is a <blank> or when the first character on the second line of the pair is a), no <space>s shall be added; otherwise, one <space> shall be added following the last character of the first line. Extra <blank>s at the start of a line shall be discarded.

Appending a character ! to the join command name shall cause a simpler join with no white-space processing, independent of the current locale.

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

5.10.7.2.13 `list`

Synopsis : [*range*] l[ist] [*count*] [*flags*]

Write the addressed lines in a way that should be unambiguous: nonprintable characters shall be written as implementation-defined multicharacter sequences; the end of the line shall be marked with a `$`.

Long lines shall be folded; the length at which folding occurs is unspecified, but should be appropriate for the output device. The only useful flag is #, for line numbers. The current line indicator shall be set to the last line written.

5.10.7.2.14 `map`

Synopsis : `map[!]` [*x rhs*]

Define macros for use in visual mode. The first argument shall be a single character. When this character is typed in visual mode, the action shall be as if the corresponding *rhs* had been typed. If the character `!` is appended to the command name `map`, the mapping shall be effective during input mode rather than command mode. This allows *x* to have two different macro definitions at the same time: one for command mode and one for input mode. Nonprintable characters, except for `<tab>`, require escaping with a `control-V` (or `control-Q`) to be entered in the arguments. On certain block-mode terminals, the mapping need not occur immediately (for example, it may occur after the terminal transmits a group of characters to the system), but it shall achieve the same results of modifying the file as if it occurred immediately. Implementations may restrict the set of commands accepted within *rhs* ; the list of restrictions is implementation defined.

The `map` command with no arguments shall write all of the macros currently defined. If `!` is appended to the command, only the macros effective during input mode shall be written; otherwise, only the macros effective during command mode shall be written.

5.10.7.2.15 `mark`

Synopsis : [*line*] ma[rk] *x*
Synopsis : [*line*] k *x*

Give the specified line the specified mark *x*, which shall be a single lowercase letter of the POSIX Locale. The current line position shall not be affected. The expression `'`*x* can then be used as an address in any command requiring one. For example ‘‘`.,'`*x*`d`’’ shall delete all the lines from the current one to the marked line. Also see 5.35.7.1.23 and 5.35.7.1.24 for uses of the mark in visual mode. If the `'`*x* command is used in nonvisual mode, the character marked shall be the first `<non->`blank character of the current line. Otherwise, the character marked shall be the character at the current column of the current line.

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

1861 #### 5.10.7.2.16 move

1862 *Synopsis* : [*range*] m[ove] *line*

1863 Move the specified lines (*range*) to be after the target line (*line*). The current line
1864 indicator shall be set to the first of the moved lines.

1865 #### 5.10.7.2.17 next

1866 *Synopsis* : n[ext][!] [*file* ...]

1867 Edit the next file from the argument list. If the current buffer has been modified
1868 since the last write, a warning shall be written and the command shall be
1869 aborted. This action can be overridden by appending the character ! to the com-
1870 mand name (n!). The argument list can be replaced by specifying a new one as
1871 operands to this command. Editing then starts with the first file on this new list.
1872 The current line indicator shall be reset as described for the edit command. This
1873 command shall be affected by the editor options autowrite and writeany; see
1874 5.10.7.5 for details.

1875 #### 5.10.7.2.18 number

1876 *Synopsis* : [*range*] nu[mber] [*count*] [*flags*]
1877 *Synopsis* : [*range*] # [*count*] [*flags*]

1878 Write the selected lines, each preceded with its line number in decimal. Non-
1879 printable characters, except for <tab>, shall be expanded as specified by the 7
1880 print command. See 5.10.7.2.21. The format shall be as follows:

1881 "%+6d\t%s", <*line number*> , <*line text*>

1882 The only meaningful flag is l, which shall cause the additional expanded writing
1883 of <tab>s and end-of-lines done by the list command to be performed. The
1884 current line indicator shall be set to the last line written.

1885 #### 5.10.7.2.19 open

1886 *Synopsis* : [*line*] o[pen] /*pattern* / [*flags*]

1887 Enter open mode, which shall be equivalent to visual mode with a one-line win-
1888 dow. All the visual mode commands shall be available. If a match is found for
1889 the optional regular expression in *line* , the cursor shall be placed at the start of
1890 the matching pattern. The visual mode command Q (see 5.35.7.1.71) shall exit
1891 open mode. This command need not be supported on block-mode terminals. 7

1892 #### 5.10.7.2.20 preserve

1893 *Synopsis* : pre[serve]

1894 Save the current buffer in a form that can later be recovered by using ex −r or by
1895 using the recover command. After the file has been preserved, a mail message
1896 shall be sent to the user. This message shall be readable by invoking the mailx
1897 utility (see 4.40). The message shall contain the name of the file, the time of

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

preservation, and an `ex` command that could be used to recover the file. Additional information may be included in the mail message.

5.10.7.2.21 `print`

Synopsis : [*range*] p[rint] [*count*] [*flags*]

Write the addressed lines. Nonprintable characters, except for <tab>, shall be written as implementation-defined multicharacter sequences. 7

Long lines shall be folded; the length at which folding occurs is unspecified, but should be appropriate for the output device. The only meaningful *flags* are # and `l`. The current line indicator shall be set to the last line written.

5.10.7.2.22 `put`

Synopsis : [*line*] pu[t] [*buffer*]

Put back deleted or ''yanked'' lines after line *line* . A buffer can be specified; otherwise, the text in the unnamed buffer (where deleted or yanked text is placed by default) shall be restored. The current line indicator shall be set to the first line put back.

5.10.7.2.23 `quit`

Synopsis : q[uit][!]

Terminate the editing session. If the current buffer has been modified since the last write, a warning shall be written and the command shall fail. This warning can be overridden, and an exit forced, discarding changes, by appending the character ! to the command name.

5.10.7.2.24 `read`

Synopsis : [*line*] r[ead][!] [*file*]

Place a copy of the specified file in the current buffer after the target line (which can be line 0 to place text at the beginning). If no *file* is named, the current file shall be the default. If there is no current file, then *file* shall become the current file. If there is no current file nor *file* operand, the command shall fail.

The current line indicator shall be set to the last line read. In visual mode, the current line indicator shall be set to the first line read. If *file* is preceded by !, *file* shall be taken to be an operating system command and passed to the program named in the **SHELL** environment variable; the resultant output shall be read in to the buffer. The special meaning of ! can be overridden by escaping it with a backslash character.

5.10.7.2.25 `recover`

Synopsis : rec[over] *file*

Attempt to recover *file* if it was saved as the result of a `preserve` command, the receipt of a signal (see 5.10.5.4), or a system or editor crash. The current line

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

indicator shall be reset as described for the `read` editor command.

5.10.7.2.26 `rewind`

Synopsis : `rew[ind][!]`

Rewind the argument list; that is, the current file shall be set to the first file in the argument list. This shall be equivalent to a `next` command with the current argument list as its operands. If the current buffer has been modified since the last write, a warning shall be written and the command shall be aborted. The action can be overridden by appending the character `!` to the command name (`rew!`). The current line indicator shall be reset as described for the `read` editor command. This command shall be affected by the editor options `autowrite` and `writeany`; see 5.10.7.5 for details.

5.10.7.2.27 `set`

Synopsis : `se[t]` [*option* [`=`[*value*]] . . .] [`no`*option* . . .] [*option* `?` . . .] [`all`]

When no arguments are specified, write those options whose values have been changed from the default settings; when the argument `all` is specified, write all of the option values.

Giving an option name followed by the character `?` shall cause the current value of that option to be written. The `?` can be separated from the option name by zero or more `<blank>`s. The `?` shall be necessary only for Boolean valued options. Boolean options can be given values by the form `se` *option* to turn them on or `se no`*option* to turn them off; string and numeric options can be assigned by the form `se` *option* `=`*value* . Any `<blank>`s in strings can be included as is by preceding each such character with a backslash. More than one option can be set or listed by a single `set` command by specifying multiple arguments, each separated from the next by one or more `<blank>`s.

See 5.10.7.5 for further details about options.

5.10.7.2.28 `shell`

Synopsis : `sh[ell]`

Invoke the program named in the **SHELL** environment variable with the argument `−i` (interactive mode). Editing shall be resumed when the program exits.

5.10.7.2.29 `source`

Synopsis : `so[urce]` *file*

Read and execute commands from the file specified by the mandatory argument *file* . Such `so` commands can be nested. The maximum supported nesting depth is implementation defined, but shall be at least one.

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

5.10.7.2.30 substitute

Synopsis : [*range*] s[ubstitute] [*/pattern /repl /*[*options*] [*count*] [*flags*]] 7

Replace the first instance of the pattern *pattern* by the string *repl* on each specified line. (See 5.10.7.3 and 5.10.7.4.) If the */pattern /repl /* argument is not present, the */pattern /repl /* from the previous `substitute` command shall be used. If *options* includes the letter `g` (global), all nonoverlapping instances of the pattern in the line shall be substituted. If the option letter `c` (confirm) is included, then before each substitution the line shall be written with `^` characters written on the following line, adjacent to and identifying the pattern to be replaced; an affirmative response shall cause the substitution to be done, while any other input aborts it. An affirmative response shall consist of a line with the affirmative response (as defined by the current locale) at the beginning of the line. Such a line shall be subject to editing in the same way as the command line (the `/` or `:` line at the bottom of the screen). The current line indicator shall be set to the last line substituted. When the `c` option is used, typing the interrupt character or receiving the SIGINT signal shall stop the substitute operation and `ex` shall return to command mode. All substitutions completed before the interrupt occurred shall be retained, and none shall be made after that point. The current line indicator shall be set to the last line substituted. This command shall be affected by the **LC_MESSAGES** environment variable and the `wrapscan` option.

5.10.7.2.31 suspend

Synopsis : su[spend][!]
Synopsis : st[op][!]

Allow control to return to the invoking process; `ex` shall suspend itself as if it had received the SIGTSTP signal. The suspension shall occur only if the system supports job control as described in POSIX .1 {8} and job control is enabled in the invoking shell (see the description of `set –m` in 3.14.11).

If job control is not supported or is disabled in the invoking shell, issuing either the `suspend` or `stop` command shall produce a warning message, but otherwise shall have no effect. Following either `suspend` or `stop` with the character `!` shall affect the operation of the `autowrite` editor option for this command only.

The current *susp* character (see `stty` in 4.59) shall also cause the suspension.

5.10.7.2.32 tag

Synopsis : ta[g][!] *tagstring*

Search for the *tagstring* , which can be in a different file. If the tag is in a different file, then the new file shall be opened for editing. If the current buffer has been modified since the last write, a warning shall be written and the command shall be aborted. The action can be overridden by appending the character `!` to the command name. The current line indicator shall be reset to the line indicated by the tag. This command shall be affected by the editor options `autowrite` and `writeany`; see 5.10.7.5 and 5.7 (`ctags`) for details. This command shall be affected by the `tags` editor option.

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

The `tag` command shall search for *tagstring* in the tag file referred to by the `tags` editor option until a reference to *tagstring* is found. The file pointed to by this reference shall be loaded into the buffer, and the current line shall be set to the first occurrence of the pattern specified in the tags file associated with the supplied *tagstring* ; if the tags file contained a line-number reference, the current line shall be set to that line. If the pattern or line number is not found, an error message shall be written. If a file referred to by the `tags` editor option does not exist or is not readable, an error message shall be written. The results are unspecified if the format of a tags file is not as specified by the `ctags` utility description.

5.10.7.2.33 `unabbrev`

Synopsis : `una[bbrev]` *word*

Delete *word* from the list of abbreviations, as described by the `abbrev` editor command in 5.10.7.2.1.

5.10.7.2.34 `undo`

Synopsis : `u[ndo]`

Reverse the changes made by the previous editing command (one that changes the contents of the buffer). For this purpose, `global` and `visual` are considered single commands. An `undo` can itself be reversed. Commands that affect the external environment, such as `write`, `edit`, and `next` cannot be undone.

5.10.7.2.35 `unmap`

Synopsis : `unm[ap][!]` *x*

If no `!` is specified, remove the command-mode macro definition for *x*; otherwise, remove the input-mode macro definition for *x*. See 5.10.7.2.14.

5.10.7.2.36 `visual`

Synopsis : [*line*] `vi[sual]` [*type*] [*count*] [*flags*]

Enter visual mode with the current line indicator set to *line* . The *type* is optional, and can be –, `.`, `+`, or `^`, as in the `z` command, to specify the position of the specified line on the screen window. (The default shall be to place the line at the top of the screen window.) A *count* specifies the number of lines that will initially be written; the default shall be the value of the editor option `window`. The command `Q` shall exit visual mode. (For more information, see `vi` in 5.35). This command need not be supported on block-mode terminals.

5.10.7.2.37 `write`

Synopsis : [*range*] `w[rite][!] [>>]` [*file*]
Synopsis : [*range*] `w[rite] [!]` [*file*]
Synopsis : [*range*] `wq[!] [>>]` [*file*]

Write the specified lines (the whole buffer, if no *range* is given) out to the file represented by the pathname *file* , writing to standard output the number of lines

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

and bytes written.

If *file* is specified and is not the current file, and the file named by *file* exists, then the write shall fail. If the current file has been changed by the `file` command, and that file exists, the write shall fail. In either case, the write can be forced by appending the character `!` to the command name. An existing file can be appended to by appending `>>` to the command name. If the file does not exist, the result is implementation defined. 7

If the *file* is preceded by `!`, the program named in the **SHELL** environment variable shall be invoked with *file* as its second argument, and the specified lines shall be passed as standard input to the command. The `!` in this usage shall be separated from the `w` command by at least one `<blank>` character. The special meaning of the `!` can be overridden by escaping it with a backslash character. This command shall be affected by the editor options `writeany` and `readonly`.

The command `wq` shall be equivalent to a `w` followed by a `q`; `wq!` shall be equivalent to `w!` followed by `q`. If the current buffer has no pathname associated with it, the `write` command shall fail.

5.10.7.2.38 `xit`

Synopsis : [*range*] x[it][!] [*file*]

Perform a `write` command (see 5.10.7.2.37) if any changes have been made to the current buffer since the last write to any file.

Unless the command fails because an attempt to write lines to a file did not succeed, the `ex` utility shall exit after an `x` command. This command shall be affected by the editor options `writeany` and `readonly`.

5.10.7.2.39 `yank`

Synopsis : [*range*] ya[nk] [*buffer*] [*count*]

Place the specified lines in the named buffer. If no buffer is specified, the unnamed buffer shall be used (where the most recently deleted or yanked text is placed by default).

5.10.7.2.40 Adjust Window

Synopsis : [*line*] z [*type*] [*count*] [*flags*]

If *type* is omitted, then *count* lines following the specified line (*line*) shall be written. The default for *count* shall be the value of the editor option `window`. The *type* argument shall change the position at which *line* shall be written on the screen by affecting the number of lines written before and after *line* .

If *type* is specified, it shall be one of the following:

- `-` Place *line* at the bottom of the screen.
- `+` Place *line* at the top of the screen.

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

`.` Place *line* in the middle.

`^` Write out *count* lines starting *count* ∗2 lines before the addressed line; the net effect of this shall be that a `z^` command following another `z` command writes the previous page.

`=` Center the addressed line on the screen with a line of hyphens written immediately before and after it. The number of preceding and following lines of text written shall be reduced to account for these lines of hyphens.

In all cases the current line indicator shall be set to the last line written, with the exception of the `=` type, which shall cause the current line indicator to be set to that addressed in the command.

5.10.7.2.41 Escape

Synopsis : `!` *command*

Synopsis : **[***range* **]**`!` *command*

Pass the remainder of the line after the `!` character to the program named in the **SHELL** environment variable for execution. A warning shall be issued if the buffer has been changed since the last write. A single `!` character shall be written when the command completes. The current line position shall not be affected.

Within the text of *command* , `%` and `#` shall be expanded as pathnames (the current and alternate pathnames, respectively), and `!` shall be replaced with the text of the previous `!` command. (Thus, `!!` shall repeat the previous `!` command.) If any such expansion is done, the expanded line shall be echoed.

The special meanings of %, #, and `!` can be overridden by escaping them with a backslash character. This command shall be affected by the editor options `autowrite` and `writeany`; see 5.10.7.5 for details.

In the second form of the `!` command, the remainder of the line after the `!` shall be passed to the program named in the **SHELL** environment variable, as described above. The specified lines shall be provided to the program as standard input; the resulting output shall replace the specified lines.

5.10.7.2.42 Shift Left

Synopsis : **[***range* **]** `<` **[***count* **]** **[***flags* **]**

Shift the specified lines to the left; the number of character positions to be shifted shall be determined by the editor option `shiftwidth`. Only leading `<blank>`s shall be lost in shifting; other characters shall not be affected. The current line indicator shall be set to the last line changed.

5.10.7.2.43 Shift Right

Synopsis : **[***range* **]** `>` **[***count* **]** **[***flags* **]**

Shift the specified lines to the right, by inserting `<blank>`s, using `<tab>`s where possible, as determined by the editor option `shiftwidth`. Empty lines shall not 7

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

be changed. The current line indicator shall be set to the last line changed.

5.10.7.2.44 Resubstitute

Synopsis : [*range*] & [*options*] [*count*] [*flags*]
Synopsis : [*range*] s[ubstitute] [*options*] [*count*] [*flags*]
Synopsis : [*range*] ~ [*options*] [*count*] [*flags*]

Repeat the previous substitute command, as if & were replaced by the previous `s/`*pattern* `/`*repl*`/` command. (The same effect can obtained by omitting the `/`*pattern* `/`*repl*`/` string in the `substitute` command.) The version of the command using tilde shall be the same as & and `s`, but the *pattern* used shall be the last regular expression used in any command, not necessarily the one used in the last substitute command. For example, in the sequence

```
s/red/blue/
/green
~
```

the ~ is equivalent to

```
s/green/blue/
```

5.10.7.2.45 Scroll

Synopsis : *eof*

Write the next *n* lines, where *n* is the value of the editor option `scroll`. The command is invoked with the *eof* character (see the description of the `stty eof` character in 4.59). The current line indicator shall be set to the last line written. This command need not be supported on block-mode terminals.

5.10.7.2.46 Write Line Number

Synopsis : [*line*] = [*flags*]

Write the line number of the specified line (default last line). The current line position shall not be affected.

5.10.7.2.47 Execute

Synopsis : @ *buffer*
Synopsis : * *buffer*

Execute each line of the named buffer as an `ex` command. If no buffer is specified or is specified as @ or *, the last buffer executed shall be used. If there is no last buffer, an error occurs.

5.10.7.3 Regular Expressions

The `ex` utility shall support the basic regular expressions described in 2.8.3. A null RE (`//`) shall be equivalent to the last RE encountered.

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

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

The following constructs can be used to enhance the basic regular expressions:

`\ <` Match the beginning of a *word*. (See the definition of *word* at the beginning of 5.10.7.2.)

`\ >` Match the end of a *word*.

`~` Match the replacement part of the last `substitute` command. The tilde (~) character can be escaped in a regular expression to become a normal character with no special meaning.

When the editor option `nomagic` is set, the only characters with special meanings shall be `^` at the beginning of a pattern, `$` at the end of a pattern, and `\`. The characters `.`, `*`, `[`, and `~` shall be treated as ordinary characters unless preceded by a `\`; when preceded by a `\` they shall regain their special meaning.

5.10.7.4 Replacement Strings

The character `&` (`\&` if the editor option `nomagic` is set) in the replacement string shall stand for the text matched by the pattern to be replaced. The character `~` (`\~` if `nomagic` is set) shall be replaced by the replacement part of the previous substitute command. The sequence `\`*n*, where *n* is an integer, shall be replaced by the text matched by the pattern enclosed in the *n*th set of parentheses `\(` and `\)`.

The strings `\l`, `\u`, `\L`, and `\U` can be used to modify the case of elements in the replacement string (using the `\&` or `\`*<digit>*) notation. The string `\l` (`\u`) shall cause the first character (actually inserted by the substitution) that follows the `\l` (`\u`) to be converted to lowercase (uppercase). The strings `\L` (`\U`) shall cause all characters subsequent to them to be converted to lowercase (uppercase) as they are inserted by the substitution until the string `\e` or `\E`, or the end of the replacement string, is encountered.

In visual mode, a `control-Vcontrol-M` (or `control-Qcontrol-M`) sequence in the replacement string shall be mapped to a `<newline>`, and so can be used to split lines. A literal `control-M` requires escaping by preceding it with a backslash (`\^V^M` or `\^Q^M`).

5.10.7.5 Edit Options

The `ex` utility has a number of options that modify its behavior. These options have default settings, which can be changed using the `set` command.

Options are Boolean unless otherwise specified.

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

5.10.7.5.1 autoindent, ai

[Default *off*]

If `autoindent` is set, each line in input mode shall be indented (using first as many `<tab>`s as possible, as determined by the editor option `tabstop`, and then using `<space>`s) to align with the previous line. (Starting indentation shall be determined by the line appended after, or the line inserted before, or the first line changed, with an `a`, `i`, or `c` command, respectively.) When a `<newline>` character is inserted in the middle of a line, and when `autoindent` is on, the first non-`<blank>` character to the right of the cursor shall be aligned to the current margin on a new line immediately following the current line. Any `<blank>`s to the left of the cursor position at which the `<newline>` was entered shall be retained. When `autoindent` is off, a new line shall be created, but no `<blanks>` shall be discarded. Additional indentation can be provided as usual; succeeding lines shall automatically be indented to the new alignment. A line entered during input mode with `autoindent` that contains no user-entered characters shall be empty, despite the appearance of indentation during entry.

Reducing the indent can be achieved when the cursor is at the current left margin by typing `control-D` one or more times; the cursor shall be moved back to the previous integral number of `shiftwidth` spaces for each `control-D`. A `^` followed by a `control-D` shall remove all indentation temporarily (for the current line); a `0` followed by a `control-D` shall remove all indentation permanently (for the current line and subsequent lines until input mode is reentered or until the indentation is specifically set to some other value). Changing the indent with `control-D` need not be supported on block-mode terminals.

5.10.7.5.2 autoprint, ap

[Default *on*]

If `autoprint` is set, the current line shall be written after each command that changes buffer text. (Autoprint shall be suppressed in the `global` [`g` and `v`] commands and for any command on which print operands [*flags*] are used to write explicitly the current line.)

5.10.7.5.3 autowrite, aw

[Default *off*]

If `autowrite` is set, when a `next`, `rewind`, `tag`, `edit`, `suspend`, `stop`, or `!` command is given, the buffer shall be written (to the current file) if it has been modified. Appending the character `!` to the command name for any of these commands except `!` shall cause the write not to occur. If the write fails, the command shall be aborted and an error message shall be written.

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

5.10.7.5.4 `beautify, bf`

[Default *off*]

If `beautify` is set, all nonprintable characters, other than `<tab>`, `<newline>`, and `<form-feed>`, shall be discarded from text read in from files.

5.10.7.5.5 `errorbells, eb`

[Default *off*]

If `errorbells` is set, error messages shall be preceded by an alert action. Setting this option off shall cause the user to be informed of an error even when in visual mode, but rather than using the `<alert>` character, an error message shall be written, using a standout mode of the terminal (such as inverse video) instead of the the normal effect of the `<alert>` character, when the effect of the `<alert>` character is to cause the terminal to ring a bell or make other sounds. The editor should place the error message in a standout mode of the terminal, such as inverse video instead of ringing the bell, when the terminal capabilities allow this.

5.10.7.5.6 `exrc`

[Default *off*]

If `exrc` is set, `ex` shall access any `.exrc` file in the current directory, as described previously. If `exrc` is not set, `ex` shall ignore any `.exrc` file in the current directory during initialization, unless the current directory is that named by the **HOME** variable.

5.10.7.5.7 `ignorecase, ic`

[Default *off*]

If `ignorecase` is set, characters that have uppercase and lowercase representations shall have those representations considered as equivalent for purposes of regular expression comparison.

5.10.7.5.8 `list`

[Default *off*]

If `list` is set, write the addressed lines in a way that should be unambiguous: nonprintable characters shall be written as implementation-defined multicharacter sequences; the end of the line shall be marked with a `$`.

5.10.7.5.9 `magic`

[Default *on*]

If `magic` is set, change the interpretation of characters in regular expressions and substitution replacement strings (see 5.10.7.3 and 5.10.7.4).

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

5.10.7.5.10 mesg

[Default *on*]

If `mesg` is set, the permission for others to use the `write` or `talk` commands to write to the terminal shall be turned on while in visual mode. The shell-level command `mesg n` (see 5.17) shall take precedence over any setting of the `ex mesg` option; i.e., if `mesg y` was issued before `ex` started (or in a shell escape, such as "`:!mesg y`"), the `mesg` option in `ex` can suppress incoming messages, but the `mesg` option cannot enable incoming messages if `mesg n` was issued.

5.10.7.5.11 number, nu

[Default *off*]

If `number` is set, lines shall be written with line numbers, as does the `number` command.

5.10.7.5.12 paragraphs, para

[Default *implementation defined*]

The `paragraph` option shall define additional paragraph boundaries for the `{` and `}` commands in visual mode. The `paragraph` option can be set to a character string consisting of zero or more character pairs. The default value is implementation defined.

In the text to be edited, the character string `<newline>.`*`<char-pair>`*, where *`<char-pair>`* is one of the character pairs found in `paragraph`, shall define a paragraph boundary. For example, if

```
paragraph=LaA ##
```

then all of the following additional paragraph boundaries would be recognized:

```
<newline>.La
<newline>.A<space>
<newline>.##
```

5.10.7.5.13 prompt

[Default *on*]

If `prompt` is set, command mode input shall be prompted for with a colon (`:`); when unset, no prompt shall be written.

5.10.7.5.14 readonly

[Default *see text*]

If `readonly` is set, read-only mode shall be enabled. Writing to a different file shall be allowed in read-only mode; in addition, the write can be forced by using the character `!` (see the editor command `write`). The default setting shall be *off* unless the file lacks write permission or the command-line option `-R` is used.

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

2304 #### 5.10.7.5.15 `remap`

2305 [Default *on*]

2306 If `remap` is set, macro translation shall allow for macros defined in terms of other
2307 macros; translation shall continue until the final product is obtained. If unset,
2308 only a one-step translation shall be done.

2309 #### 5.10.7.5.16 `report`

2310 [Default 5]

2311 The value of this option shall give the number of lines that can be changed by an
2312 editor command before a report is generated on the number of lines affected.

2313 #### 5.10.7.5.17 `scroll`

2314 [Default `window`/2]

2315 The value of this option shall determine the number of lines scrolled on a *eof* com-
2316 mand (see 5.10.7.2.45 and the description of the `stty eof` character in 4.59). 8
2317 Changing the value of `window` in **EXINIT**, one of the `.exrc` files, or with a `set` 8
2318 command shall not affect the value of `scroll`. This editor option need not be 8
2319 supported on block-mode terminals.

2320 #### 5.10.7.5.18 `sections`

2321 [Default *implementation defined*]

2322 The `sections` option shall define additional section boundaries for the `[[` and `]]`
2323 commands in visual mode. The `sections` option can be set to a character string
2324 consisting of zero or more character pairs. The default value is implementation
2325 defined.

2326 In the text to be edited, the character string `<newline>.`*<char-pair>*, where
2327 *<char-pair>* is one of the character pairs found in `sections`, shall define a sec-
2328 tion boundary in the same manner that paragraph boundaries are defined. (See
2329 5.10.7.5.12.)

2330 #### 5.10.7.5.19 `shell, sh`

2331 [Default from the environment **SHELL**]

2332 The value of this option can be a string representing the pathname of the shell to
2333 be invoked for the `!` shell escape command, and by the `shell` command. The
2334 default shall be taken from the **SHELL** variable in the environment; see 5.10.5.3
2335 for default values for **SHELL**.

2336 #### 5.10.7.5.20 `shiftwidth, sw`

2337 [Default 8]

2338 The value of this option shall give the width of an indentation level used during
2339 `autoindent` and by the shift commands.

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

5.10.7.5.21 showmatch, sm

[Default *off*]

If `showmatch` is set, in visual mode, when a `)` or `}` is typed, the matching `(` or `{` shall be shown if it is still on the screen. This editor option need not be supported on block-mode terminals.

5.10.7.5.22 showmode

[Default *off*]

If `showmode` is set, in visual mode, the current mode that the editor is in shall be written on the last line of the screen. Modes that shall be reported are command mode and input mode; other unspecified modes may be written.

5.10.7.5.23 tabstop, ts

[Default 8]

The value of this option shall specify the software tab stops to be used by the editor to expand tabs in the input.

5.10.7.5.24 tags

[Default *see text*] 7

The value of this option can be a string representing `<space>`-separated pathnames that shall be used as tag files for the `tag` command. A requested tag shall be searched for sequentially in the specified files. By default, filenames of `tags` shall be searched for in the current directory and in other implementation-defined directories.

5.10.7.5.25 term

[Default from the environment **TERM**]

The value of this option can be a string representing the terminal type of the output device. The default shall be taken from the **TERM** variable in the environment; see 5.10.5.3 for default values for **TERM** .

5.10.7.5.26 terse

[Default *off*]

If `terse` is set, error messages may be less verbose. However, except for this caveat, error messages are unspecified. Furthermore, not all error messages need change for different settings of this option.

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

2371 #### 5.10.7.5.27 warn

2372 [Default *on*]

2373 If warn is set, ex shall write a warning message to standard error if the contents
2374 of the buffer have not been saved before a ! command escape.

2375 #### 5.10.7.5.28 window

2376 [Default *see text*]

2377 The value of this option shall determine the default number of lines in a screen-
2378 ful, as written by the z command. When in visual mode, the number of lines out-
2379 put when moving up or down the file by a *screenful* (see 5.35). The value of win-
2380 dow can be unrelated to the real screen size, except that it shall be set on entry to
2381 be the current number of screen lines. (The current number of screen lines shall 7
2382 be determined by the system or overridden by the user, as described for **LINES** in 7
2383 5.10.5.3 and 2.6.) The baud rate of the terminal line may reduce the default in an 7
2384 implementation-defined manner. The default value of windows also can be over- 7
2385 ridden by specifying a window size using the –w command-line option. 7

2386 #### 5.10.7.5.29 wrapscan, ws

2387 [Default *on*]

2388 If wrapscan is set, searches (using // or ??) shall wrap around the end of the
2389 editing buffer; when unset, searches shall stop at the beginning of the editing
2390 buffer for ??, or at the end of the editing buffer for //.

2391 #### 5.10.7.5.30 wrapmargin, wm

2392 [Default 0]

2393 If the value of this option is greater than zero (say *n*) in visual mode during text
2394 entry, then, in the POSIX Locale, a <newline> shall replace all consecutive 7
2395 <blank>s, at the boundary between a <blank> and a non-<blank>, so that lines 7
2396 shall end at least *n* spaces from the ending margin of the terminal screen. (The 7
2397 ending margin shall be determined by the system or overridden by the user, as 7
2398 described for **COLUMNS** in 5.10.5.3 and 2.6.) If a line consists of a sequence of 7
2399 <non->blank characters long enough such that it extends continuously from the
2400 beginning margin to beyond the ending margin, that sequence shall not be broken
2401 by the action of this option. If the value is zero, no wrapping shall be performed.

2402 #### 5.10.7.5.31 writeany, wa

2403 [Default *off*]

2404 If writeany is set, file-overwriting checks shall be inhibited that would otherwise 7
2405 be made before write and xit commands, or before an automatic write (see edi-
2406 tor option autowrite), allowing a write to any file (provided permissions allow
2407 it).

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

5.10.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.10.9 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, ex shall terminate with a nonzero exit status.

— If the command in error was one of those specified with the command-line −c option (or the obsolescent + option), all of the remaining −c commands shall be flushed, and a new command read from standard input.

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

Usage, Examples

All of the following are valid *addresses* :

`+++` three lines after the current line

`/`*re*`/-` one line before the next occurrence of *re*

`-2` two lines before the current line

The rules for case-insensitive matching (editor option `ignorecase`) may lead to counter-intuitive situations when uppercase characters are used in range expressions. Historical versions of this text implied that the process was:

(1) Take a line of text from buffer

(2) Convert uppercase to lowercase in text line

(3) Convert uppercase to lowercase in RE, except in character class specifications

(4) Match RE against text

This would mean that, with `ignorecase` in effect, the text

```
The cat sat on the mat
```

would be matched by

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

```
/^the/
```

but not by

```
/^[A-Z]he/
```

If a SIGSEGV signal is received while `ex` is saving a file, the file might not be successfully saved.

The `next` command can accept more than one file, so usage such as

```
next 'ls [abc]*'
```

is valid; it would not be valid for the `edit` or `read` commands, for example, because they expect only one file and unspecified results occur.

An example of case conversion with the `s` command:

```
:p
The cat sat on the mat.
:s/\<.at\>/\u&/gp
The Cat Sat on the Mat.
:s/S\(.*\)M/S\U\1\eM/p
The Cat SAT ON THE Mat.
```

The following table is a condensed version of information contained in the normative text. It is presented here to facilitate the review for accuracy of the `ex` commands and the editor options that affect those commands.

ex Command	**Editor Option**	**Description**
a[ppend]	autoindent	Determines starting indentation by the line immediately preceding the append.
a[ppend]!	autoindent	a! causes the autoindent editor option setting to be toggled for the duration of this command only.
c[hange]	autoindent	Determines starting indentation based on the first line that was changed.
c[hange]!	autoindent	c! causes the autoindent editor option setting to be toggled for the duration of this command only.
e[dit] ex	autowrite	Causes the buffer to be written to the current file if it has been modified. (e! or ex! causes write not to occur.)
	writeany	Inhibits checks before an automatic write, allowing a write to any file (provided that permissions allow it).
g[lobal] v	autoindent	Editor option is inhibited when using this command.
	autoprint	Editor option is inhibited when using this command.

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

	`ignorecase`	Characters that have uppercase and lowercase representations shall have those representations considered as equivalent for purposes of regular expression comparison.
	`magic`	When set, the characters `^ $ . * [ ~` have special meaning in regular expressions. When unset, only characters `^ $` have special meaning in regular expressions.
	`report`	Editor option is inhibited when using this command.
`i[nsert]`	`autoindent`	Determines starting indentation based on the line that is inserted before.
`i[nsert]!`	`autoindent`	`i!` causes the autoindent editor option setting to be toggled for the duration of this command only.
`n[ext]`	`autowrite`	Causes the buffer to be written to the current file if it has been modified. (`n!` causes write not to occur.)
	`writeany`	Inhibits checks before an automatic write, allowing a write to any file (provided that permissions allow it).
`p[rint]`	`list`	Causes nonprintable characters to be written, and marks the end of line with $. 7 7
	`number`	Causes lines to be written with a line number before the text.
`rew[ind]`	`autowrite`	Causes the buffer to be written to the current file if it has been modified. (`rew!` causes write not to occur.)
	`writeany`	Inhibits checks before an automatic write, allowing a write to any file (provided that permissions allow it).
`sh[ell]`	`shell`	The pathname of the shell to be invoked.
`s[ubstitute]`	`ignorecase`	Characters that have uppercase and lowercase representations shall have those representations considered as equivalent for purposes of regular expression comparison.
	`magic`	When set, the characters `^ $ . * [ ~` have special meaning in regular expressions. When unset, only characters `^ $` have special meaning in regular expressions.
	`report`	Determines the number of lines that can be changed by an editor command before a report is generated.
	`wrapscan`	When set, searches (using `//` or `??`) shall wrap

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

		around the end of the file. When unset, searches shall stop at the beginning of the file for ??, or at the end of the file for //.
su[spend] st[op]	autowrite	Causes the buffer to be written to the current file if it has been modified. (su! or st! causes write not to occur.)
ta[g]	autowrite	Causes the buffer to be written to the current file if it has been modified. (ta! causes write not to occur.)
	tags	File searched when looking for a tag.
	writeany	Inhibits checks before an automatic write, allowing a write to any file (provided that permissions allow it).
vi[sual]	window	Default window size used when invoking vi command.
w[rite]	readonly	Prohibits write to file unless w! is entered.
	writeany	Inhibits checks before a write, allowing a write to any file (provided that permissions allow it).
x[it]	readonly	Prohibits write to file unless x! is entered.
	writeany	Inhibits checks before an exit, allowing a write to any file (provided that permissions allow it).
z	window	Default screen size if not explicitly stated.
! (Escape)	autowrite	Causes the buffer to be written to the current file if it has been modified.
	shell	The pathname of the shell to be invoked.
	warn	Warn user before a ! command escape if there has been ''[No write since last change]''
	writeany	Inhibits checks before an automatic write, allowing a write to any file (provided that permissions allow it).
<	shiftwidth	Determines the number of character positions to be shifted.
>	shiftwidth	Determines the number of character positions to be shifted.
eof	scroll	Determines the number of lines to scroll.

The −s option (and its obsolescent single-hyphen form) suppresses all interactive user feedback and is useful for editing scripts in batch jobs. The list of specific effects is derived from historical practice. The terminal type ''incapable of supporting visual mode'' historically has been named dumb. 8

The uses described for control-V can also be accomplished with control-Q, which is useful on terminals that use control-V for the down-arrow function. However, most historical implementations use control-Q for the *termios* START character, so the editor will generally not receive the control-Q unless stty −ixon mode is set. Any of the command characters described in this standard 8

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

can be made ineffective by their selection as *termios* control characters, using the `stty` utility or other methods described in POSIX.1 {8}. 8 8

History of Decisions Made

It is recognized that `ex` visual mode (`vi`) and related features would be difficult, if not impossible, to implement satisfactorily on a synchronous terminal, thus it is not a mandatory requirement that such features should work on all terminals. It is the intention, however, that an `ex` implementation should provide the full set of capabilities on all terminals capable of supporting them.

A conforming syntax has been provided for `ex`.

A ''restricted editor'' (both the historical `red` utility and modifications to `ex`) were considered and rejected for inclusion. Neither option provided the level of security that many users might expect.

The `-c` replacement for ''+command'' was inspired by the `-e` option of `sed`.

The `-l` command for LISP mode was omitted as it was considered that it would be difficult to justify the inclusion of such programming language-dependent features for one language and not for others. Similarly the `lisp` edit mode option was omitted.

The `-t` option was introduced because the `ctags` command appears in POSIX.2 and the option is already widely available.

The `-x` option for encryption, and the associated `crypt` utility, were omitted because the algorithm used was not specifiable and the export control laws of some nations make it difficult to export cryptographic technology.

It was agreed that the ability to edit files of arbitrary type can be occasionally useful, but it was not felt necessary to mandate that an `ex` implementation be required to handle other than text files.

The paragraph in Input Files, ''By default, ...,'' is intended to close a long-standing security hole in `ex` and `vi`, that of the little-known ''modelines.'' This ''feature'' is that whereby any line in the first or last five lines of the file containing the strings `"ex:"` or `"vi:"` (and, apparently `"ei:"` or `"vx:"`) is considered to be a line containing editor commands and `ex` interprets all the text up to the next `:` or `<newline>` as a command. Consider the consequences, for example, of an unsuspecting user using `ex` or `vi` as their editor when replying to a mail message in which a line such as `ex:! rm -rf *:` appeared in the signature lines. It was strongly felt that an editor should not *by default* interpret any lines of a file. If this feature is desired it should be under the user's control either by the use of a command-line option or by placing some control line in a initialization file. This approach has indeed already been adopted by some vendors (HP-UX has the `modelines` option, for instance, which is turned off by default).

The current documentation for `ex` seems to imply a particular implementation of a scheme for recovering files after some ''disaster.'' Such details were eliminated.

Explicit dependence on the ASCII character set has been avoided where possible, hence the use of an ''implementation-defined multicharacter sequence'' for the

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

display of nonprintable characters in preference to the historical usage of, say, ^I for <tab>. Room is left for an implementation to select some other means of displaying characters unambiguously: combinations of octal or hexadecimal values with leading slashes along with the escape sequences in Table 2-13. (In fact, the historical ^I is not really unambiguous at all, because it could actually be the characters ^ and I; allowing this historical practice is why the word ''should'' is used.) It is recommended that if an implementation chooses a new method for this writing that it match the formats used by ed and sed in POSIX. 2.

The writing of NUL by the p command is not specified as a special case because the group did not want to require ex to support NUL characters in files and buffers.

The action of the join command in relation to its action with white space is only defined for the POSIX Locale as the correct amount of white space after a period varies; in Japanese none is required, in French only a single space, and so on.

References to ''function keys'' in the map command were omitted because the specification was too simple to be generally useful in a portable manner.

The version command was not included because no sufficiently general specification of the version information could be devised that would still be useful to a portable user. This command name should be used by suppliers who wish to provide version information about the ex and vi editors.

The directory editor option was omitted because it was implementation-specific.

The edcompatible editor option was omitted because it was not (ed-compatible).

The default for editor option exrc was reversed from common practice because it was perceived as a security hole by both the POSIX. 2 and POSIX. 6 working groups. The POSIX. 2 group does not believe this to be an undue burden because those people who use .exrc file in directories other than $HOME can retain the old behavior (by setting this option in their **EXINIT** environment variables or their $HOME/.exrc files). Note that noexrc disables the use of ./.exrc; exrc enables it. The default is to disable it.

The paragraph and section options are in the standard, even though their historical defaults are very oriented to the UNIX system troff text formatter, because a ''portable user'' could use the {, }, [[, or]] commands in visual mode and have the cursor stop in unexpected places.

The redraw and slowopen options were omitted because they are implementation dependent.

In historical implementations the default for editor option window is arrived at by the following steps:

(1) The initial default is half the number of lines on the terminal screen as defined in a terminal database (typically, *terminfo* or *termcap*).

(2) If dynamic screen size information is available either from environment variables (System V) or the TTY driver (BSD), this information is used

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

instead.

(3) If the number of lines is erroneous, the default is set to 23.

(4) If the baud rate is less than 2400, the default is set to 16.

(5) If the baud rate is less than 1200, the default is set to 8. This default can then be changed by the editor options below:

`window` set the window size explicitly and always

`w300` set `window` if baud rate is less than 1200

`w1200` set `window` if baud rate is less than 2400

`w9600` set `window` if baud rate is greater than 2400

The `w300`, `w1200`, and `w9600` options do not appear in this standard because of their lack of documentation and dependence on specific baud rates.

The action taken for signals other than SIGINT , SIGCONT , and SIGHUP is unspecified because some implementations attempt to save the editing buffer in a useful state when other signals are received.

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

5.11 expand — Convert tabs to spaces

5.11.1 Synopsis

`expand [–t` *tablist* `] [`*file* `...]`

Obsolescent Version :

`expand [-`*tabstop* `] [-`*tab1* `,`*tab2* `,...,`*tabn* `] [`*file* `...]`

5.11.2 Description

The `expand` utility shall write files or the standard input to the standard output with `<tab>` characters replaced with one or more `<space>` characters needed to pad to the next tab stop. Any `<backspace>` characters shall be copied to the output and cause the column position count for tab stop calculations to be decremented; the column position count shall not be decremented below zero.

5.11.3 Options

The `expand` utility shall conform to the utility argument syntax guidelines described in 2.10.2; the obsolescent version does not.

The following options shall be supported by the implementation:

`–t` *tablist* Specify the tab stops. The argument *tablist* shall consist of a single positive decimal integer or multiple positive decimal integers, separated by `<blank>`s or commas, in ascending order. If a single number is given, tabs shall be set *tablist* column positions apart instead of the default 8. If multiple numbers are given, the tabs shall be set at those specific column positions.

Each tab-stop position *N* shall be an integer value greater than zero, and the list shall be in strictly ascending order. This shall be taken to mean that, from the start of a line of output, tabbing to position *N* shall cause the next character output to be in the (*N*+1)th column position on that line.

In the event of `expand` having to process a `<tab>` character at a position beyond the last of those specified in a multiple tab-stop list, the `<tab>` shall be replaced by a single `<space>` in the output.

In the obsolescent version, the single number is specified as *tabstop* with a leading minus; multiple tab stops are specified after a leading minus as *tab1* , *tab2* , etc.

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