Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'ex.1p' command

$ man ex.1p

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

PROLOG

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

NAME

ex ? text editor

SYNOPSIS

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

DESCRIPTION

The ex utility is a line-oriented text editor. There are two other modes of the editor?open and visual?in which screen-oriented editing is available. This is described more fully by the ex open and visual com? mands and in vi.

If an operand is '-', the results are unspecified.

This section uses the term edit buffer to describe the current working text. No specific implementation is implied by this term. All editing changes are performed on the edit buffer, and no changes to it shall affect any file until an editor command writes the file.

Certain 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'' or 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 volume of POSIX.1?2017 or report an error describing the terminal-related deficiency.

OPTIONS

The ex utility shall conform to the Base Definitions volume of POSIX.1?2017, Section 12.2, Utility Syntax Guidelines, except for the unspecified usage of '-', and that '+' may be recognized as an option delimiter as well as '-'.

The following options shall be supported:

-c command

Specify an initial command to be executed in the first edit buffer loaded from an existing file (see the EXTENDED DESCRIPTION section). Implementations may support more than a single -c option. In such implementations, the specified commands shall be executed in the order specified on the command line.

-r Recover the named files (see the EXTENDED DESCRIPTION section). Recovery information for a file shall be saved during an editor or system crash (for example, when the editor is terminated by a signal which the editor can catch), or after the use of an ex preserve 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 number of changes to the edit buffer (since the most recent preserve command) that will be recovered is unspecified.

If no file operands are given and the -t option is not specified, 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 the edi? tor shall exit normally without further action.

-R Set readonly edit option.

-s Prepare ex for batch use by taking the following actions:

* Suppress writing prompts and informational (but not diag? nostic) messages.
* Ignore the value of TERM and any implementation default terminal type and assume the terminal is a type incapable of supporting open or visual modes; see the visual com? mand and the description of vi.
* Suppress the use of the EXINIT environment variable and the reading of any .exrc file; see the EXTENDED DESCRIP? TION section.
* Suppress autoindentation, ignoring the value of the au? toindent edit option.

-t tagstring

Edit the file containing the specified tagstring; see ctags. The tags feature represented by -t tagstring and the tag com? mand is optional. It shall be provided on any system that also provides a conforming implementation of ctags; other? wise, the use of -t produces undefined results. On any sys? tem, it shall be an error to specify more than a single -t option.

-v Begin in visual mode (see vi).

-w size Set the value of the window editor option to size.

OPERANDS

The following operand shall be supported:

file A pathname of a file to be edited.

STDIN

The standard input consists of a series of commands and input text, as described in the EXTENDED DESCRIPTION section. The implementation may limit each line of standard input to a length of {LINE_MAX}.

If the standard input is not a terminal device, it shall be as if the -s option had been specified.

If a read from the standard input returns an error, or if the editor detects an end-of-file condition from the standard input, it shall be equivalent to a SIGHUP asynchronous event.

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. By default, any incomplete last line shall be treated as if it had a trailing <newline>.

The editing of other forms of files may optionally be allowed by implementations.

The .exrc files and source files shall be text files consisting of ex commands; see the EXTENDED DESCRIPTION section.

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

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of ex:

COLUMNS Override the system-selected horizontal screen size. See the Base Definitions volume of POSIX.1?2017, Chapter 8, Environment Variables for valid values and results when it is unset or null.

EXINIT Determine a list of ex commands that are executed on editor start-up. See the EXTENDED DESCRIPTION section for more details of the initialization phase.

HOME Determine a pathname of a directory that shall be searched for an editor start-up file named .exrc; see the EXTENDED DESCRIPTION section.

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

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

LC_COLLATE

Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements within regu? lar expressions.

LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files), the behavior of character classes within regular ex? pressions, the classification of characters as uppercase or lowercase letters, the case conversion of letters, and the detection of word boundaries.

LC_MESSAGES

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

LINES 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 the Base Definitions volume of POSIX.1?2017, Chapter 8, Environment Variables for valid val? ues and results when it is unset or null.

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

PATH Determine the search path for the shell command specified in the ex editor commands !, shell, read, and write, and the open and visual mode command !; see the description of com? mand search and execution in Section 2.9.1.1, Command Search and Execution.

SHELL Determine the preferred command line interpreter for use as the default value of the shell edit option.

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

used.

ASYNCHRONOUS EVENTS

The following term is used in this and following sections to specify command and asynchronous event actions:

complete write

A complete write is a write of the entire contents of the edit buffer to a file of a type other than a terminal device, or the saving of the edit buffer caused by the user executing the ex preserve command. Writing the contents of the edit buffer to a temporary file that will be removed when the edi? tor exits shall not be considered a complete write.

The following actions shall be taken upon receipt of signals:

SIGINT If the standard input is not a terminal device, ex shall not write the file or return to command or text input mode, and shall exit with a non-zero exit status.

Otherwise, if executing an open or visual text input mode command, ex in receipt of SIGINT shall behave identically to its receipt of the <ESC> character.

Otherwise:

1. If executing an ex text input mode command, all input lines that have been completely entered shall be resolved into the edit buffer, and any partially entered line shall be discarded.
2. If there is a currently executing command, it shall be aborted and a message displayed. Unless otherwise speci? fied by the ex or vi command descriptions, it is unspeci? fied whether any lines modified by the executing command appear modified, or as they were before being modified by the executing command, in the buffer.

   If the currently executing command was a motion command, its associated command shall be discarded.
3. If in open or visual command mode, the terminal shall be alerted.

4. The editor shall then return to command mode.

SIGCONT The screen shall be refreshed if in open or visual mode.

SIGHUP If the edit buffer has been modified since the last complete write, ex shall attempt to save the edit buffer so that it can be recovered later using the -r option or the ex recover command. The editor shall not write the file or return to command or text input mode, and shall terminate with a non-zero exit status.

SIGTERM Refer to SIGHUP.

The action taken for all other signals is unspecified.

STDOUT

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

STDERR

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

The output from ex shall be text files.

EXTENDED DESCRIPTION

Only the ex mode of the editor is described in this section. See vi for additional editing capabilities available in ex.

When an error occurs, ex shall write a message. If the terminal sup? ports a standout mode (such as inverse video), the message shall be written in standout mode. If the terminal does not support a standout mode, and the edit option errorbells is set, an alert action shall pre? cede the error message.

By default, ex shall start in command mode, which shall be indicated by a : prompt; see the prompt command. Text input mode can be entered by the 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.

Initialization in ex and vi

The following symbols are used in this and following sections to spec? ify locations in the edit buffer:

alternate and current pathnames

Two pathnames, named current and alternate, are maintained by the editor. Any ex commands that take filenames as arguments shall set them as follows:

1. If a file argument is specified to the ex edit, ex, or re? cover commands, or if an ex tag command replaces the contents of the edit buffer.
   a. If the command replaces the contents of the edit buffer, the current pathname shall be set to the file argument or the file indicated by the tag, and the alternate pathname shall be set to the previous value of the current path? name.
   b. Otherwise, the alternate pathname shall be set to the file argument.
2. If a file argument is specified to the ex next command:
   a. If the command replaces the contents of the edit buffer, the current pathname shall be set to the first file argu? ment, and the alternate pathname shall be set to the pre? vious value of the current pathname.
3. If a file argument is specified to the ex file command, the current pathname shall be set to the file argument, and the alternate pathname shall be set to the previous value of the current pathname.
4. If a file argument is specified to the ex read and write com? mands (that is, when reading or writing a file, and not to the program named by the shell edit option), or a file argu? ment is specified to the ex xit command:
   a. If the current pathname has no value, the current path? name shall be set to the file argument.
   b. Otherwise, the alternate pathname shall be set to the file argument.

If the alternate pathname is set to the previous value of the current pathname when the current pathname had no previous value,

then the alternate pathname shall have no value as a result.

current line

The line of the edit buffer referenced by the cursor. Each com? mand description specifies the current line after the command has been executed, as the current line value. When the edit buffer contains no lines, the current line shall be zero; see Addressing in ex.

current column

The current display line column occupied by the cursor. (The col? umns shall be numbered beginning at 1.) Each command description specifies the current column after the command has been executed, as the current column value. This column is an ideal column that is remembered over the lifetime of the editor. The actual display line column upon which the cursor rests may be different from the current column; see the cursor positioning discussion in Command Descriptions in vi.

set to non-<blank>

A description for a current column value, meaning that the cur? rent column shall be set to the last display line column on which is displayed any part of the first non-<blank> of the line. If the line has no non-<blank> non-<newline> characters, the current column shall be set to the last display line column on which is displayed any part of the last non-<newline> character in the line. If the line is empty, the current column shall be set to column position 1.

The length of lines in the edit buffer may be limited to {LINE_MAX} bytes. In open and visual mode, the length of lines in the edit buffer may be limited to the number of characters that will fit in the dis? play. If either limit is exceeded during editing, an error message shall be written. If either limit is exceeded by a line read in from a file, an error message shall be written and the edit session may be terminated.

If the editor stops running due to any reason other than a user com?

mand, and the edit buffer has been modified since the last complete write, it shall be equivalent to a SIGHUP asynchronous event. If the system crashes, it shall be equivalent to a SIGHUP asynchronous event.

During initialization (before the first file is copied into the edit buffer or any user commands from the terminal are processed) the fol? lowing shall occur:

1. If the environment variable EXINIT is set, the editor shall execute the ex commands contained in that variable.
2. If the EXINIT variable is not set, and all of the following are true:
   a. The HOME environment variable is not null and not empty.
   b. The file .exrc in the directory referred to by the HOME envi? ronment variable:
      i. Exists
      ii. Is owned by the same user ID as the real user ID of the process or the process has appropriate privileges
      iii. Is not writable by anyone other than the owner

   the editor shall execute the ex commands contained in that file.
3. If and only if all of the following are true:
   a. The current directory is not referred to by the HOME environ? ment variable.
   b. A command in the EXINIT environment variable or a command in the .exrc file in the directory referred to by the HOME envi? ronment variable sets the editor option exrc.
   c. The .exrc file in the current directory:
      i. Exists
      ii. Is owned by the same user ID as the real user ID of the process, or by one of a set of implementation-defined user IDs
      iii. Is not writable by anyone other than the owner

   the editor shall attempt to execute the ex commands contained in that file.

Lines in any .exrc file that are blank lines shall be ignored. If any

.exrc file exists, but is not read for ownership or permission reasons, it shall be an error.

After the EXINIT variable and any .exrc files are processed, the first file specified by the user shall be edited, as follows:

1. If the user specified the -t option, the effect shall be as if the ex tag command was entered with the specified argument, with the exception that if tag processing does not result in a file to edit, the effect shall be as described in step 3. below.
2. Otherwise, if the user specified any command line file arguments, the effect shall be as if the ex edit command was entered with the first of those arguments as its file argument.
3. Otherwise, the effect shall be as if the ex edit command was en? tered with a nonexistent filename as its file argument. It is un? specified whether this action shall set the current pathname. In an implementation where this action does not set the current pathname, any editor command using the current pathname shall fail until an editor command sets the current pathname.

If the -r option was specified, the first time a file in the initial argument list or a file specified by the -t option is edited, if recov? ery information has previously been saved about it, that information shall be recovered and the editor shall behave as if the contents of the edit buffer have already been modified. If there are multiple in? stances of the file to be recovered, the one most recently saved shall be recovered, and an informational message that there are previous ver? sions of the file that can be recovered shall be written. If no recov? ery information about a file is available, an informational message to this effect shall be written, and the edit shall proceed as usual.

If the -c option was specified, the first time a file that already ex? ists (including a file that might not exist but for which recovery in? formation is available, when the -r option is specified) replaces or initializes the contents of the edit buffer, the current line shall be set to the last line of the edit buffer, the current column shall be set to non-<blank>, and the ex commands specified with the -c option

shall be executed. In this case, the current line and current column shall not be set as described for the command associated with the re? placement or initialization of the edit buffer contents. However, if the -t option or a tag command is associated with this action, the -c option commands shall be executed and then the movement to the tag shall be performed.

The current argument list shall initially be set to the filenames spec? ified by the user on the command line. If no filenames are specified by the user, the current argument list shall be empty. If the -t option was specified, it is unspecified whether any filename resulting from tag processing shall be prepended to the current argument list. In the case where the filename is added as a prefix to the current argument list, the current argument list reference shall be set to that file? name. In the case where the filename is not added as a prefix to the current argument list, the current argument list reference shall logi? cally be located before the first of the filenames specified on the command line (for example, a subsequent ex next command shall edit the first filename from the command line). If the -t option was not speci? fied, the current argument list reference shall be to the first of the filenames on the command line.

Addressing in ex

Addressing in ex relates to the current line and the current column; the address of a line is its 1-based line number, the address of a col? umn is its 1-based count from the beginning of the line. Generally, the current line is the last line affected by a command. The current line number is the address of the current line. In each command description, the effect of the command on the current line number and the current column is described.

Addresses are constructed as follows:

1. The character '.' (period) shall address the current line.
2. The character '$' shall address the last line of the edit buffer.
3. The positive decimal number n shall address the nth line of the edit buffer.

4. The address "'x" refers to the line marked with the mark name char?
acter 'x', which shall be a lowercase letter from the portable
character set, the backquote character, or the single-quote charac?
ter. It shall be an error if the line that was marked is not cur?
rently present in the edit buffer or the mark has not been set.
Lines can be marked with the ex mark or k commands, or the vi m
command.

5. A regular expression enclosed by <slash> characters ('/') shall ad?
dress the first line found by searching forwards from the line fol?
lowing the current line toward the end of the edit buffer and stop?
ping at the first line for which the line excluding the terminating
<newline> matches the regular expression. As stated in Regular Ex?
pressions in ex, an address consisting of a null regular expression
delimited by <slash> characters ("//") shall address the next line
for which the line excluding the terminating <newline> matches the
last regular expression encountered. In addition, the second
<slash> can be omitted at the end of a command line. If the wrap?
scan edit option is set, the search shall wrap around to the begin?
ning of the edit buffer and continue up to and including the cur?
rent line, so that the entire edit buffer is searched. Within the
regular expression, the sequence "\/" shall represent a literal
<slash> instead of the regular expression delimiter.

6. A regular expression enclosed in <question-mark> characters ('?')
shall address the first line found by searching backwards from the
line preceding the current line toward the beginning of the edit
buffer and stopping at the first line for which the line excluding
the terminating <newline> matches the regular expression. An ad?
dress consisting of a null regular expression delimited by <ques?
tion-mark> characters ("??") shall address the previous line for
which the line excluding the terminating <newline> matches the last
regular expression encountered. In addition, the second <question-
mark> can be omitted at the end of a command line. If the wrapscan
edit option is set, the search shall wrap around from the beginning

of the edit buffer to the end of the edit buffer and continue up to and including the current line, so that the entire edit buffer is searched. Within the regular expression, the sequence "\?" shall represent a literal <question-mark> instead of the RE delimiter.

7. A <plus-sign> ('+') or a <hyphen-minus> ('-') followed by a decimal number shall address the current line plus or minus the number. A '+' or '-' not followed by a decimal number shall address the cur? rent line plus or minus 1.

Addresses can be followed by zero or more address offsets, optionally <blank>-separated. Address offsets are constructed as follows:

1. A '+' or '-' immediately followed by a decimal number shall add (subtract) the indicated number of lines to (from) the address. A '+' or '-' not followed by a decimal number shall add (subtract) 1 to (from) the address.
2. A decimal number shall add the indicated number of lines to the ad? dress.

It shall not be an error for an intermediate address value to be less than zero or greater than the last line in the edit buffer. It shall be an error for the final address value to be less than zero or greater than the last line in the edit buffer.

Commands take zero, one, or two addresses; see the descriptions of 1addr and 2addr in Command Descriptions in ex. If more than the re? quired number of addresses are provided to a command that requires zero addresses, it shall be an error. Otherwise, if more than the required number of addresses are provided to a command, the addresses specified first shall be evaluated and then discarded until the maximum number of valid addresses remain.

Addresses shall be separated from each other by a <comma> (',') or a <semicolon> (';'). If no address is specified before or after a <comma> or <semicolon> separator, it shall be as if the address of the current line was specified before or after the separator. In the case of a <semicolon> separator, the current line ('.') shall be set to the first address, and only then will the next address be calculated. This

feature can be used to determine the starting line for forwards and backwards searches (see rules 5. and 6.).

A <percent-sign> ('%') shall be equivalent to entering the two addresses "1,$".

Any delimiting <blank> characters between addresses, address separators, or address offsets shall be discarded.

Command Line Parsing in ex

The following symbol is used in this and following sections to describe parsing behavior:

escape If a character is referred to as ``<backslash>-escaped'' or ``<control>?V-escaped'', it shall mean that the character acquired or lost a special meaning by virtue of being preceded, respectively, by a <backslash> or <control>?V character. Unless otherwise specified, the escaping character shall be discarded at that time and shall not be further considered for any purpose.

Command-line parsing shall be done in the following steps. For each step, characters already evaluated shall be ignored; that is, the phrase ``leading character'' refers to the next character that has not yet been evaluated.

1. Leading <colon> characters shall be skipped.
2. Leading <blank> characters shall be skipped.
3. If the leading character is a double-quote character, the characters up to and including the next non-<backslash>-escaped <newline> shall be discarded, and any subsequent characters shall be parsed as a separate command.
4. Leading characters that can be interpreted as addresses shall be evaluated; see Addressing in ex.
5. Leading <blank> characters shall be skipped.
6. If the next character is a <vertical-line> character or a <newline>:
   a. If the next character is a <newline>:
      i. If ex is in open or visual mode, the current line shall be

set to the last address specified, if any.

ii. Otherwise, if the last command was terminated by a <verti? cal-line> character, no action shall be taken; for exam? ple, the command "||<newline>" shall execute two implied commands, not three.

iii. Otherwise, step 6.b. shall apply.

b. Otherwise, the implied command shall be the print command. The last #, p, and l flags specified to any ex command shall be re? membered and shall apply to this implied command. Executing the ex number, print, or list command shall set the remembered flags to #, nothing, and l, respectively, plus any other flags specified for that execution of the number, print, or list com? mand.

If ex is not currently performing a global or v command, and no address or count is specified, the current line shall be incre? mented by 1 before the command is executed. If incrementing the current line would result in an address past the last line in the edit buffer, the command shall fail, and the increment shall not happen.

c. The <newline> or <vertical-line> character shall be discarded and any subsequent characters shall be parsed as a separate command.

7. The command name shall be comprised of the next character (if the character is not alphabetic), or the next character and any subse? quent alphabetic characters (if the character is alphabetic), with the following exceptions:

a. Commands that consist of any prefix of the characters in the command name delete, followed immediately by any of the charac? ters 'l', 'p', '+', '-', or '#' shall be interpreted as a delete command, followed by a <blank>, followed by the charac? ters that were not part of the prefix of the delete command. The maximum number of characters shall be matched to the com? mand name delete; for example, "del" shall not be treated as

"de" followed by the flag l.

b. Commands that consist of the character 'k', followed by a char? acter that can be used as the name of a mark, shall be equiva? lent to the mark command followed by a <blank>, followed by the character that followed the 'k'.

c. Commands that consist of the character 's', followed by charac? ters that could be interpreted as valid options to the s com? mand, shall be the equivalent of the s command, without any pattern or replacement values, followed by a <blank>, followed by the characters after the 's'.

8. The command name shall be matched against the possible command names, and a command name that contains a prefix matching the char? acters specified by the user shall be the executed command. In the case of commands where the characters specified by the user could be ambiguous, the executed command shall be as follows:

```
??????????????????????????????????????
?a  ? append ??n  ? next  ??t  ? t     ?
?c  ? change ??p  ? print ??u  ? undo  ?
?ch ? change ??pr ? print ??un ? undo  ?
?e  ? edit   ??r  ? read  ??v  ? v     ?
?m  ? move   ??re ? read  ??w  ? write ?
?ma ? mark   ??s  ? s     ??   ?       ?
??????????????????????????????????????
```

Implementation extensions with names causing similar ambiguities shall not be checked for a match until all possible matches for commands specified by POSIX.1?2008 have been checked.

9. If the command is a ! command, or if the command is a read command followed by zero or more <blank> characters and a !, or if the com? mand is a write command followed by one or more <blank> characters and a !, the rest of the command shall include all characters up to a non-<backslash>-escaped <newline>. The <newline> shall be dis? carded and any subsequent characters shall be parsed as a separate ex command.

10. Otherwise, if the command is an edit, ex, or next command, or a visual command while in open or visual mode, the next part of the command shall be parsed as follows:

a. Any '!' character immediately following the command shall be skipped and be part of the command.

b. Any leading <blank> characters shall be skipped and be part of the command.

c. If the next character is a '+', characters up to the first non-<backslash>-escaped <newline> or non-<backslash>-escaped <blank> shall be skipped and be part of the command.

d. The rest of the command shall be determined by the steps speci? fied in paragraph 12.

11. Otherwise, if the command is a global, open, s, or v command, the next part of the command shall be parsed as follows:

a. Any leading <blank> characters shall be skipped and be part of the command.

b. If the next character is not an alphanumeric, double-quote, <newline>, <backslash>, or <vertical-line> character:

i. The next character shall be used as a command delimiter.

ii. If the command is a global, open, or v command, characters up to the first non-<backslash>-escaped <newline>, or first non-<backslash>-escaped delimiter character, shall be skipped and be part of the command.

iii. If the command is an s command, characters up to the first non-<backslash>-escaped <newline>, or second non-<back? slash>-escaped delimiter character, shall be skipped and be part of the command.

c. If the command is a global or v command, characters up to the first non-<backslash>-escaped <newline> shall be skipped and be part of the command.

d. Otherwise, the rest of the command shall be determined by the steps specified in paragraph 12.

12. Otherwise:

a. If the command was a map, unmap, abbreviate, or unabbreviate command, characters up to the first non-<control>?V-escaped <newline>, <vertical-line>, or double-quote character shall be skipped and be part of the command.

b. Otherwise, characters up to the first non-<backslash>-escaped <newline>, <vertical-line>, or double-quote character shall be skipped and be part of the command.

c. If the command was an append, change, or insert command, and the step 12.b. ended at a <vertical-line> character, any subse? quent characters, up to the next non-<backslash>-escaped <new? line> shall be used as input text to the command.

d. If the command was ended by a double-quote character, all sub? sequent characters, up to the next non-<backslash>-escaped <newline>, shall be discarded.

e. The terminating <newline> or <vertical-line> character shall be discarded and any subsequent characters shall be parsed as a separate ex command.

Command arguments shall be parsed as described by the Synopsis and De? scription of each individual ex command. This parsing shall not be <blank>-sensitive, except for the ! argument, which must follow the command name without intervening <blank> characters, and where it would otherwise be ambiguous. For example, count and flag arguments need not be <blank>-separated because "d22p" is not ambiguous, but file argu? ments to the ex next command must be separated by one or more <blank> characters. Any <blank> in command arguments for the abbreviate, unab? breviate, map, and unmap commands can be <control>?V-escaped, in which case the <blank> shall not be used as an argument delimiter. Any <blank> in the command argument for any other command can be <back? slash>-escaped, in which case that <blank> shall not be used as an ar? gument delimiter.

Within command arguments for the abbreviate, unabbreviate, map, and un? map commands, any character can be <control>?V-escaped. All such es? caped characters shall be treated literally and shall have no special

meaning. Within command arguments for all other ex commands that are not regular expressions or replacement strings, any character that would otherwise have a special meaning can be <backslash>-escaped. Es? caped characters shall be treated literally, without special meaning as shell expansion characters or '!', '%', and '#' expansion characters.

See Regular Expressions in ex and Replacement Strings in ex for de? scriptions of command arguments that are regular expressions or re? placement strings.

Non-<backslash>-escaped '%' characters appearing in file arguments to any ex command shall be replaced by the current pathname; unescaped '#' characters shall be replaced by the alternate pathname. It shall be an error if '%' or '#' characters appear unescaped in an argument and their corresponding values are not set.

Non-<backslash>-escaped '!' characters in the arguments to either the ex ! command or the open and visual mode ! command, or in the argu? ments to the ex read command, where the first non-<blank> after the command name is a '!' character, or in the arguments to the ex write command where the command name is followed by one or more <blank> char? acters and the first non-<blank> after the command name is a '!' char? acter, shall be replaced with the arguments to the last of those three commands as they appeared after all unescaped '%', '#', and '! char? acters were replaced. It shall be an error if '!' characters appear unescaped in one of these commands and there has been no previous exe? cution of one of these commands.

If an error occurs during the parsing or execution of an ex command:

* An informational message to this effect shall be written. Execution of the ex command shall stop, and the cursor (for example, the cur? rent line and column) shall not be further modified.
* If the ex command resulted from a map expansion, all characters from that map expansion shall be discarded, except as otherwise specified by the map command.
* Otherwise, if the ex command resulted from the processing of an EX? INIT environment variable, a .exrc file, a :source command, a -c

option, or a +command specified to an ex edit, ex, next, or visual command, no further commands from the source of the commands shall be executed.

* Otherwise, if the ex command resulted from the execution of a buf? fer or a global or v command, no further commands caused by the ex? ecution of the buffer or the global or v command shall be executed.

* Otherwise, if the ex command was not terminated by a <newline>, all characters up to and including the next non-<backslash>-escaped <newline> shall be discarded.

Input Editing in ex

The following symbol is used in this and the following sections to specify command actions:

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 edit buffer.

When accepting input characters from the user, in either ex command mode or ex text input mode, ex shall enable canonical mode input pro? cessing, as defined in the System Interfaces volume of POSIX.1?2017.

If in ex text input mode:

1. If the number edit option is set, ex shall prompt for input using the line number that would be assigned to the line if it is en? tered, in the format specified for the ex number command.
2. If the autoindent edit option is set, ex shall prompt for input us? ing autoindent characters, as described by the autoindent edit op? tion. autoindent characters shall follow the line number, if any.

If in ex command mode:

1. If the prompt edit option is set, input shall be prompted for using a single ':' character; otherwise, there shall be no prompt.

The input characters in the following sections shall have the following effects on the input line.

Scroll

Synopsis:

eof

See the description of the stty eof character in stty.

If in ex command mode:

If the eof character is the first character entered on the line, the line shall be evaluated as if it contained two characters: a <control>?D and a <newline>.

Otherwise, the eof character shall have no special meaning.

If in ex text input mode:

If the cursor follows an autoindent character, the autoindent characters in the line shall be modified so that a part of the next text input character will be displayed on the first column in the line after the previous shiftwidth edit option column boundary, and the user shall be prompted again for input for the same line.

Otherwise, if the cursor follows a '0', which follows an autoin? dent character, and the '0' was the previous text input charac? ter, the '0' and all autoindent characters in the line shall be discarded, and the user shall be prompted again for input for the same line.

Otherwise, if the cursor follows a '^', which follows an autoin? dent character, and the '^' was the previous text input charac? ter, the '^' and all autoindent characters in the line shall be discarded, and the user shall be prompted again for input for the same line. In addition, the autoindent level for the next input line shall be derived from the same line from which the autoindent level for the current input line was derived.

Otherwise, if there are no autoindent or text input characters in the line, the eof character shall be discarded.

Otherwise, the eof character shall have no special meaning.

<newline>

Synopsis:

<newline>

<control>-J

If in ex command mode:

Cause the command line to be parsed; <control>?J shall be mapped to the <newline> for this purpose.

If in ex text input mode:

Terminate the current line. If there are no characters other than autoindent characters on the line, all characters on the line shall be discarded.

Prompt for text input on a new line after the current line. If the autoindent edit option is set, an appropriate number of au? toindent characters shall be added as a prefix to the line as described by the ex autoindent edit option.

<backslash>

Synopsis:

<backslash>

Allow the entry of a subsequent <newline> or <control>?J as a literal character, removing any special meaning that it may have to the editor during text input mode. The <backslash> character shall be retained and evaluated when the command line is parsed, or retained and included when the input text becomes part of the edit buffer.

<control>?V

Synopsis:

<control>-V

Allow the entry of any subsequent character as a literal character, re? moving any special meaning that it may have to the editor during text input mode. The <control>?V character shall be discarded before the command line is parsed or the input text becomes part of the edit buf? fer.

If the ``literal next'' functionality is performed by the underlying system, it is implementation-defined whether a character other than <control>?V performs this function.

<control>?W

Synopsis:

<control>-W

Discard the <control>?W, and the word previous to it in the input line, including any <blank> characters following the word and preceding the <control>?W. If the ``word erase'' functionality is performed by the underlying system, it is implementation-defined whether a character other than <control>?W performs this function.

Command Descriptions in ex

The following symbols are used in this section to represent command modifiers. Some of these modifiers can be omitted, in which case the specified defaults shall be used.

1addr A single line address, given in any of the forms described in Addressing in ex; the default shall be the current line ('.'), unless otherwise specified.

If the line address is zero, it shall be an error, unless otherwise specified in the following command descriptions.

If the edit buffer is empty, and the address is specified with a command other than =, append, insert, open, put, read, or visual, or the address is not zero, it shall be an error.

2addr Two addresses specifying an inclusive range of lines. If no addresses are specified, the default for 2addr shall be the current line only (".,."), unless otherwise specified in the following command descriptions. If one address is specified, 2addr shall specify that line only, unless otherwise speci? fied in the following command descriptions.

It shall be an error if the first address is greater than the second address.

If the edit buffer is empty, and the two addresses are speci? fied with a command other than the !, write, wq, or xit com? mands, or either address is not zero, it shall be an error.

count A positive decimal number. If count is specified, it shall be equivalent to specifying an additional address to the com? mand, unless otherwise specified by the following command de? scriptions. The additional address shall be equal to the last address specified to the command (either explicitly or by de?

fault) plus count-1.

If this would result in an address greater than the last line of the edit buffer, it shall be corrected to equal the last line of the edit buffer.

flags One or more of the characters '+', '-', '#', 'p', or 'l' (ell). The flag characters can be <blank>-separated, and in any order or combination. The characters '#', 'p', and 'l' shall cause lines to be written in the format specified by the print command with the specified flags.

The lines to be written are as follows:

1. All edit buffer lines written during the execution of the ex &, ~, list, number, open, print, s, visual, and z com? mands shall be written as specified by flags.
2. After the completion of an ex command with a flag as an argument, the current line shall be written as specified by flags, unless the current line was the last line writ? ten by the command.

The characters '+' and '-' cause the value of the current line after the execution of the ex command to be adjusted by the offset address as described in Addressing in ex. This adjustment shall occur before the current line is written as described in 2. above.

The default for flags shall be none.

buffer One of a number of named areas for holding text. The named buffers are specified by the alphanumeric characters of the POSIX locale. There shall also be one ``unnamed'' buffer. When no buffer is specified for editor commands that use a buffer, the unnamed buffer shall be used. Commands that store text into buffers shall store the text as it was before the command took effect, and shall store text occurring ear? lier in the file before text occurring later in the file, re? gardless of how the text region was specified. Commands that store text into buffers shall store the text into the unnamed

buffer as well as any specified buffer.

In ex commands, buffer names are specified as the name by it? self. In open or visual mode commands the name is preceded by a double-quote ("") character.

If the specified buffer name is an uppercase character, and the buffer contents are to be modified, the buffer shall be appended to rather than being overwritten. If the buffer is not being modified, specifying the buffer name in lowercase and uppercase shall have identical results.

There shall also be buffers named by the numbers 1 through 9.

In open and visual mode, if a region of text including char? acters from more than a single line is being modified by the vi c or d commands, the motion character associated with the c or d commands specifies that the buffer text shall be in line mode, or the commands %, `, /, ?, (,), N, n, {, or } are used to define a region of text for the c or d commands, the contents of buffers 1 through 8 shall be moved into the buffer named by the next numerically greater value, the con? tents of buffer 9 shall be discarded, and the region of text shall be copied into buffer 1. This shall be in addition to copying the text into a user-specified buffer or unnamed buf? fer, or both. Numeric buffers can be specified as a source buffer for open and visual mode commands; however, specifying a numeric buffer as the write target of an open or visual mode command shall have unspecified results.

The text of each buffer shall have the characteristic of be? ing in either line or character mode. Appending text to a non-empty buffer shall set the mode to match the characteris? tic of the text being appended. Appending text to a buffer shall cause the creation of at least one additional line in the buffer. All text stored into buffers by ex commands shall be in line mode. The ex commands that use buffers as the source of text specify individually how buffers of different

modes are handled. Each open or visual mode command that uses buffers for any purpose specifies individually the mode of the text stored into the buffer and how buffers of different modes are handled.

file Command text used to derive a pathname. The default shall be the current pathname, as defined previously, in which case, if no current pathname has yet been established it shall be an error, except where specifically noted in the individual command descriptions that follow. If the command text con? tains any of the characters '~', '{', '[', '*', '?', '$', "", backquote, single-quote, and <backslash>, it shall be subjected to the process of ``shell expansions'', as de? scribed below; if more than a single pathname results and the command expects only one, it shall be an error.

The process of shell expansions in the editor shall be done as follows. The ex utility shall pass two arguments to the program named by the shell edit option; the first shall be -c, and the second shall be the string "echo" and the command text as a single argument. The standard output and standard error of that command shall replace the command text.

! A character that can be appended to the command name to mod? ify its operation, as detailed in the individual command de? scriptions. With the exception of the ex read, write, and ! commands, the '!' character shall only act as a modifier if there are no <blank> characters between it and the command name.

remembered search direction

The vi commands N and n begin searching in a forwards or backwards direction in the edit buffer based on a remembered search direction, which is initially unset, and is set by the ex global, v, s, and tag commands, and the vi / and ? com? mands.

Synopsis:

ab[breviate][lhs rhs]

If lhs and rhs are not specified, write the current list of abbrevia?
tions and do nothing more.

Implementations may restrict the set of characters accepted in lhs or rhs, except that printable characters and <blank> characters shall not be restricted. Additional restrictions shall be implementation-defined.

In both lhs and rhs, any character may be escaped with a <control>?V, in which case the character shall not be used to delimit lhs from rhs, and the escaping <control>?V shall be discarded.

In open and visual text input mode, if a non-word or <ESC> character that is not escaped by a <control>?V character is entered after a word character, a check shall be made for a set of characters matching lhs, in the text input entered during this command. If it is found, the ef?
fect shall be as if rhs was entered instead of lhs.

The set of characters that are checked is defined as follows:

1. If there are no characters inserted before the word and non-word or <ESC> characters that triggered the check, the set of characters shall consist of the word character.
2. If the character inserted before the word and non-word or <ESC> characters that triggered the check is a word character, the set of characters shall consist of the characters inserted immediately be?
   fore the triggering characters that are word characters, plus the triggering word character.
3. If the character inserted before the word and non-word or <ESC> characters that triggered the check is not a word character, the set of characters shall consist of the characters that were in?
   serted before the triggering characters that are neither <blank> characters nor word characters, plus the triggering word character.

It is unspecified whether the lhs argument entered for the ex abbrevi?
ate and unabbreviate commands is replaced in this fashion. Regardless of whether or not the replacement occurs, the effect of the command shall be as if the replacement had not occurred.

Current line: Unchanged.

Current column: Unchanged.

Append

Synopsis:

[1addr] a[ppend][!]

Enter ex text input mode; the input text shall be placed after the specified line. If line zero is specified, the text shall be placed at the beginning of the edit buffer.

This command shall be affected by the number and autoindent edit options; following the command name with '!' shall cause the autoindent edit option setting to be toggled for the duration of this command only.

Current line: Set to the last input line; if no lines were input, set to the specified line, or to the first line of the edit buffer if a line of zero was specified, or zero if the edit buffer is empty.

Current column: Set to non-<blank>.

Arguments

Synopsis:

ar[gs]

Write the current argument list, with the current argument-list entry, if any, between '[' and ']' characters.

Current line: Unchanged.

Current column: Unchanged.

Change

Synopsis:

[2addr] c[hange][!][count]

Enter ex text input mode; the input text shall replace the specified lines. The specified lines shall be copied into the unnamed buffer, which shall become a line mode buffer.

This command shall be affected by the number and autoindent edit options; following the command name with '!' shall cause the autoindent edit option setting to be toggled for the duration of this command only.

Current line: Set to the last input line; if no lines were input, set to the line before the first address, or to the first line of the edit buffer if there are no lines preceding the first address, or to zero if the edit buffer is empty.

Current column: Set to non-<blank>.

Change Directory

Synopsis:

chd[ir][!][directory]

cd[!][directory]

Change the current working directory to directory.

If no directory argument is specified, and the HOME environment vari? able is set to a non-null and non-empty value, directory shall default to the value named in the HOME environment variable. If the HOME envi? ronment variable is empty or is undefined, the default value of direc? tory is implementation-defined.

If no '!' is appended to the command name, and the edit buffer has been modified since the last complete write, and the current pathname does not begin with a '/', it shall be an error.

Current line: Unchanged.

Current column: Unchanged.

Copy

Synopsis:

[2addr] co[py] 1addr [flags]

[2addr] t 1addr [flags]

Copy the specified lines after the specified destination line; line zero specifies that the lines shall be placed at the beginning of the edit buffer.

Current line: Set to the last line copied.

Current column: Set to non-<blank>.

Delete

Synopsis:

[2addr] d[elete][buffer][count][flags]

Delete the specified lines into a buffer (defaulting to the unnamed

buffer), which shall become a line-mode buffer.

Flags can immediately follow the command name; see Command Line Parsing in ex.

Current line: Set to the line following the deleted lines, or to the last line in the edit buffer if that line is past the end of the edit buffer, or to zero if the edit buffer is empty.

Current column: Set to non-<blank>.

Edit

Synopsis:

e[dit][!][+command][file]

ex[!][+command][file]

If no '!' is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error.

If file is specified, replace the current contents of the edit buffer with the current contents of file, and set the current pathname to file. If file is not specified, replace the current contents of the edit buffer with the current contents of the file named by the current pathname. If for any reason the current contents of the file cannot be accessed, the edit buffer shall be empty.

The +command option shall be <blank>-delimited; <blank> characters within the +command can be escaped by preceding them with a <backslash> character. The +command shall be interpreted as an ex command immedi? ately after the contents of the edit buffer have been replaced and the current line and column have been set.

If the edit buffer is empty:

Current line: Set to 0.

Current column: Set to 1.

Otherwise, if executed while in ex command mode or if the +command ar? gument is specified:

Current line: Set to the last line of the edit buffer.

Current column: Set to non-<blank>.

Otherwise, if file is omitted or results in the current pathname:

Current line: Set to the first line of the edit buffer.

Current column: Set to non-<blank>.

Otherwise, if file is the same as the last file edited, the line and column shall be set as follows; if the file was previously edited, the line and column may be set as follows:

Current line: Set to the last value held when that file was last edited. If this value is not a valid line in the new edit buffer, set to the first line of the edit buffer.

Current column: If the current line was set to the last value held when the file was last edited, set to the last value held when the file was last edited. Otherwise, or if the last value is not a valid column in the new edit buffer, set to non-<blank>.

Otherwise:

Current line: Set to the first line of the edit buffer.

Current column: Set to non-<blank>.

File

Synopsis:

f[ile][file]

If a file argument is specified, the alternate pathname shall be set to the current pathname, and the current pathname shall be set to file.

Write an informational message. If the file has a current pathname, it shall be included in this message; otherwise, the message shall indi? cate that there is no current pathname. If the edit buffer contains lines, the current line number and the number of lines in the edit buf? fer shall be included in this message; otherwise, the message shall in? dicate that the edit buffer is empty. If the edit buffer has been modi? fied since the last complete write, this fact shall be included in this message. If the readonly edit option is set, this fact shall be in? cluded in this message. The message may contain other unspecified in? formation.

Current line: Unchanged.

Current column: Unchanged.

Global

Synopsis:

[2addr] g[lobal] /pattern/ [commands]

[2addr] v /pattern/ [commands]

The optional '!' character after the global command shall be the same as executing the v command.

If pattern is empty (for example, "//") or not specified, the last regular expression used in the editor command shall be used as the pattern. The pattern can be delimited by <slash> characters (shown in the Synopsis), as well as any non-alphanumeric or non-<blank> other than <backslash>, <vertical-line>, <newline>, or double-quote.

If no lines are specified, the lines shall default to the entire file.

The global and v commands are logically two-pass operations. First, mark the lines within the specified lines for which the line excluding the terminating <newline> matches (global) or does not match (v or global!) the specified pattern. Second, execute the ex commands given by commands, with the current line ('.') set to each marked line. If an error occurs during this process, or the contents of the edit buffer are replaced (for example, by the ex :edit command) an error message shall be written and no more commands resulting from the execution of this command shall be processed.

Multiple ex commands can be specified by entering multiple commands on a single line using a <vertical-line> to delimit them, or one per line, by escaping each <newline> with a <backslash>.

If no commands are specified:

1. If in ex command mode, it shall be as if the print command were specified.
2. Otherwise, no command shall be executed.

For the append, change, and insert commands, the input text shall be included as part of the command, and the terminating <period> can be omitted if the command ends the list of commands. The open and visual commands can be specified as one of the commands, in which case each marked line shall cause the editor to enter open or visual mode. If open or visual mode is exited using the vi Q command, the current line shall be set to the next marked line, and open or visual mode reen?

tered, until the list of marked lines is exhausted.

The global, v, and undo commands cannot be used in commands. Marked lines may be deleted by commands executed for lines occurring earlier in the file than the marked lines. In this case, no commands shall be executed for the deleted lines.

If the remembered search direction is not set, the global and v com? mands shall set it to forward.

The autoprint and autoindent edit options shall be inhibited for the duration of the g or v command.

Current line: If no commands executed, set to the last marked line. Otherwise, as specified for the executed ex commands.

Current column: If no commands are executed, set to non-<blank>; other? wise, as specified for the individual ex commands.

Insert

Synopsis:

[1addr] i[nsert][!]

Enter ex text input mode; the input text shall be placed before the specified line. If the line is zero or 1, the text shall be placed at the beginning of the edit buffer.

This command shall be affected by the number and autoindent edit op? tions; following the command name with '!' shall cause the autoindent edit option setting to be toggled for the duration of this command only.

Current line: Set to the last input line; if no lines were input, set to the line before the specified line, or to the first line of the edit buffer if there are no lines preceding the specified line, or zero if the edit buffer is empty.

Current column: Set to non-<blank>.

Join

Synopsis:

[2addr] j[oin][!][count][flags]

If count is specified:

If no address was specified, the join command shall behave as if

2addr were the current line and the current line plus count (.,. + count).

If one address was specified, the join command shall behave as if 2addr were the specified address and the specified address plus count (addr,addr + count).

If two addresses were specified, the join command shall behave as if an additional address, equal to the last address plus count -1 (addr1,addr2,addr2 + count -1), was specified.

If this would result in a second address greater than the last line of the edit buffer, it shall be corrected to be equal to the last line of the edit buffer.

If no count is specified:

If no address was specified, the join command shall behave as if 2addr were the current line and the next line (.,. +1).

If one address was specified, the join command shall behave as if 2addr were the specified address and the next line (addr,addr +1).

Join the text from the specified lines together into a single line, which shall replace the specified lines.

If a '!' character is appended to the command name, the join shall be without modification of any line, independent of the current locale.

Otherwise, in the POSIX locale, set the current line to the first of the specified lines, and then, for each subsequent line, proceed as follows:

1. Discard leading <space> characters from the line to be joined.
2. If the line to be joined is now empty, delete it, and skip steps 3 through 5.
3. If the current line ends in a <blank>, or the first character of the line to be joined is a ')' character, join the lines without further modification.
4. If the last character of the current line is a '.', join the lines with two <space> characters between them.
5. Otherwise, join the lines with a single <space> between them.

Current line: Set to the first line specified.

Current column: Set to non-<blank>.

List

Synopsis:

[2addr] l[ist][count][flags]

This command shall be equivalent to the ex command:

[2addr] p[rint][count] l[flags]

See Print.

Map

Synopsis:

map[!][lhs rhs]

If lhs and rhs are not specified:

1. If '!' is specified, write the current list of text input mode maps.
2. Otherwise, write the current list of command mode maps.
3. Do nothing more.

Implementations may restrict the set of characters accepted in lhs or rhs, except that printable characters and <blank> characters shall not be restricted. Additional restrictions shall be implementation-defined.

In both lhs and rhs, any character can be escaped with a <control>?V, in which case the character shall not be used to delimit lhs from rhs, and the escaping <control>?V shall be discarded.

If the character '!' is appended to the map command name, the mapping shall be effective during open or visual text input mode rather than open or visual command mode. This allows lhs to have two different map definitions at the same time: one for command mode and one for text in? put mode.

For command mode mappings:

When the lhs is entered as any part of a vi command in open or visual mode (but not as part of the arguments to the command), the action shall be as if the corresponding rhs had been en? tered.

If any character in the command, other than the first, is es?

caped using a <control>?V character, that character shall not be part of a match to an lhs.

It is unspecified whether implementations shall support map com? mands where the lhs is more than a single character in length, where the first character of the lhs is printable.

If lhs contains more than one character and the first character is '#', followed by a sequence of digits corresponding to a num? bered function key, then when this function key is typed it shall be mapped to rhs. Characters other than digits following a '#' character also represent the function key named by the characters in the lhs following the '#' and may be mapped to rhs. It is unspecified how function keys are named or what function keys are supported.

For text input mode mappings:

When the lhs is entered as any part of text entered in open or visual text input modes, the action shall be as if the corre? sponding rhs had been entered.

If any character in the input text is escaped using a <con? trol>?V character, that character shall not be part of a match to an lhs.

It is unspecified whether the lhs text entered for subsequent map or unmap commands is replaced with the rhs text for the pur? poses of the screen display; regardless of whether or not the display appears as if the corresponding rhs text was entered, the effect of the command shall be as if the lhs text was en? tered.

If only part of the lhs is entered, it is unspecified how long the edi? tor will wait for additional, possibly matching characters before treating the already entered characters as not matching the lhs.

The rhs characters shall themselves be subject to remapping, unless otherwise specified by the remap edit option, except that if the char? acters in lhs occur as prefix characters in rhs, those characters shall not be remapped.

On block-mode terminals, the mapping need not occur immediately (for example, it may occur after the terminal transmits a group of charac? ters to the system), but it shall achieve the same results as if it oc? curred immediately.

Current line: Unchanged.

Current column: Unchanged.

Mark

Synopsis:

[1addr] ma[rk] character

[1addr] k character

Implementations shall support character values of a single lowercase letter of the POSIX locale and the backquote and single-quote charac? ters; support of other characters is implementation-defined.

If executing the vi m command, set the specified mark to the current line and 1-based numbered character referenced by the current column, if any; otherwise, column position 1.

Otherwise, set the specified mark to the specified line and 1-based numbered first non-<blank> non-<newline> in the line, if any; other? wise, the last non-<newline> in the line, if any; otherwise, column po? sition 1.

The mark shall remain associated with the line until the mark is reset or the line is deleted. If a deleted line is restored by a subsequent undo command, any marks previously associated with the line, which have not been reset, shall be restored as well. Any use of a mark not asso? ciated with a current line in the edit buffer shall be an error.

The marks ` and ' shall be set as described previously, immediately be? fore the following events occur in the editor:

1. The use of '$' as an ex address
2. The use of a positive decimal number as an ex address
3. The use of a search command as an ex address
4. The use of a mark reference as an ex address
5. The use of the following open and visual mode commands: <con? trol>?], %, (,), [,], {, }

6. The use of the following open and visual mode commands: ', G, H, L, M, z if the current line will change as a result of the command

7. The use of the open and visual mode commands: /, ?, N, `, n if the current line or column will change as a result of the command

8. The use of the ex mode commands: z, undo, global, v

For rules 1., 2., 3., and 4., the ` and ' marks shall not be set if the ex command is parsed as specified by rule 6.a. in Command Line Parsing in ex.

For rules 5., 6., and 7., the ` and ' marks shall not be set if the commands are used as motion commands in open and visual mode.

For rules 1., 2., 3., 4., 5., 6., 7., and 8., the ` and ' marks shall not be set if the command fails.

The ` and ' marks shall be set as described previously, each time the contents of the edit buffer are replaced (including the editing of the initial buffer), if in open or visual mode, or if in ex mode and the edit buffer is not empty, before any commands or movements (including commands or movements specified by the -c or -t options or the +command argument) are executed on the edit buffer. If in open or visual mode, the marks shall be set as if executing the vi m command; otherwise, as if executing the ex mark command.

When changing from ex mode to open or visual mode, if the ` and ' marks are not already set, the ` and ' marks shall be set as described previ? ously.

Current line: Unchanged.

Current column: Unchanged.

Move

Synopsis:

[2addr] m[ove] 1addr [flags]

Move the specified lines after the specified destination line. A desti? nation of line zero specifies that the lines shall be placed at the be? ginning of the edit buffer. It shall be an error if the destination line is within the range of lines to be moved.

Current line: Set to the last of the moved lines.

Current column: Set to non-<blank>.

Next

Synopsis:

n[ext][!][+command][file ...]

If no '!' is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error, unless the file is successfully written as specified by the autowrite option.

If one or more files is specified:

1. Set the argument list to the specified filenames.
2. Set the current argument list reference to be the first entry in the argument list.
3. Set the current pathname to the first filename specified.

Otherwise:

1. It shall be an error if there are no more filenames in the argument list after the filename currently referenced.
2. Set the current pathname and the current argument list reference to the filename after the filename currently referenced in the argument list.

Replace the contents of the edit buffer with the contents of the file named by the current pathname. If for any reason the contents of the file cannot be accessed, the edit buffer shall be empty.

This command shall be affected by the autowrite and writeany edit options.

The +command option shall be <blank>-delimited; <blank> characters can be escaped by preceding them with a <backslash> character. The +command shall be interpreted as an ex command immediately after the contents of the edit buffer have been replaced and the current line and column have been set.

Current line: Set as described for the edit command.

Current column: Set as described for the edit command.

Number

Synopsis:

[2addr] nu[mber][count][flags]

[2addr] #[count][flags]

These commands shall be equivalent to the ex command:

[2addr] p[rint][count] #[flags]

See Print.

Open

Synopsis:

[1addr] o[pen] /pattern/ [flags]

This command need not be supported on block-mode terminals or terminals with insufficient capabilities. If standard input, standard output, or standard error are not terminal devices, the results are unspecified.

Enter open mode.

The trailing delimiter can be omitted from pattern at the end of the command line. If pattern is empty (for example, "//") or not specified, the last regular expression used in the editor shall be used as the pattern. The pattern can be delimited by <slash> characters (shown in the Synopsis), as well as any alphanumeric, or non-<blank> other than <backslash>, <vertical-line>, <newline>, or double-quote.

Current line: Set to the specified line.

Current column: Set to non-<blank>.

Preserve

Synopsis:

pre[serve]

Save the edit buffer in a form that can later be recovered by using the -r option or by using the ex recover command. After the file has been preserved, a mail message shall be sent to the user. This message shall be readable by invoking the mailx utility. The message shall contain the name of the file, the time of preservation, and an ex command that could be used to recover the file. Additional information may be in? cluded in the mail message.

Current line: Unchanged.

Current column: Unchanged.

Print

Synopsis:

[2addr] p[rint][count][flags]

Write the addressed lines. The behavior is unspecified if the number of columns on the display is less than the number of columns required to write any single character in the lines being written.

Non-printable characters, except for the <tab>, shall be written as im? plementation-defined multi-character sequences.

If the # flag is specified or the number edit option is set, each line shall be preceded by its line number in the following format:

"%6d ", <line number>

If the l flag is specified or the list edit option is set:

1. The characters listed in the Base Definitions volume of POSIX.1?2017, Table 5-1, Escape Sequences and Associated Actions shall be written as the corresponding escape sequence.
2. Non-printable characters not in the Base Definitions volume of POSIX.1?2017, Table 5-1, Escape Sequences and Associated Actions shall be written as one three-digit octal number (with a preceding <backslash>) for each byte in the character (most significant byte first).
3. The end of each line shall be marked with a '$', and literal '$' characters within the line shall be written with a preceding <back? slash>.

Long lines shall be folded; the length at which folding occurs is un? specified, but should be appropriate for the output terminal, consider? ing the number of columns of the terminal.

If a line is folded, and the l flag is not specified and the list edit option is not set, it is unspecified whether a multi-column character at the folding position is separated; it shall not be discarded.

Current line: Set to the last written line.

Current column: Unchanged if the current line is unchanged; otherwise, set to non-<blank>.

Put

Synopsis:

[1addr] pu[t][buffer]

Append text from the specified buffer (by default, the unnamed buffer) to the specified line; line zero specifies that the text shall be placed at the beginning of the edit buffer. Each portion of a line in the buffer shall become a new line in the edit buffer, regardless of the mode of the buffer.

Current line: Set to the last line entered into the edit buffer.

Current column: Set to non-<blank>.

Quit

Synopsis:

q[uit][!]

If no '!' is appended to the command name:

1. If the edit buffer has been modified since the last complete write, it shall be an error.
2. If there are filenames in the argument list after the filename currently referenced, and the last command was not a quit, wq, xit, or ZZ (see Exit) command, it shall be an error.

Otherwise, terminate the editing session.

Read

Synopsis:

[1addr] r[ead][!][file]

If '!' is not the first non-<blank> to follow the command name, a copy of the specified file shall be appended into the edit buffer after the specified line; line zero specifies that the copy shall be placed at the beginning of the edit buffer. The number of lines and bytes read shall be written. If no file is named, the current pathname shall be the default. If there is no current pathname, then file shall become the current pathname. If there is no current pathname or file operand, it shall be an error. Specifying a file that is not of type regular shall have unspecified results.

Otherwise, if file is preceded by '!', the rest of the line after the '!' shall have '%', '#', and '!' characters expanded as described in Command Line Parsing in ex.

The ex utility shall then pass two arguments to the program named by the shell edit option; the first shall be -c and the second shall be the expanded arguments to the read command as a single argument. The standard input of the program shall be set to the standard input of the ex program when it was invoked. The standard error and standard output of the program shall be appended into the edit buffer after the speci? fied line.

Each line in the copied file or program output (as delimited by <new? line> characters or the end of the file or output if it is not immedi? ately preceded by a <newline>), shall be a separate line in the edit buffer. Any occurrences of <carriage-return> and <newline> pairs in the output shall be treated as single <newline> characters.

The special meaning of the '!' following the read command can be over? ridden by escaping it with a <backslash> character.

Current line: If no lines are added to the edit buffer, unchanged. Oth? erwise, if in open or visual mode, set to the first line entered into the edit buffer. Otherwise, set to the last line entered into the edit buffer.

Current column: Set to non-<blank>.

Recover

Synopsis:

rec[over][!] file

If no '!' is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error.

If no file operand is specified, then the current pathname shall be used. If there is no current pathname or file operand, it shall be an error.

If no recovery information has previously been saved about file, the recover command shall behave identically to the edit command, and an informational message to this effect shall be written.

Otherwise, set the current pathname to file, and replace the current contents of the edit buffer with the recovered contents of file. If there are multiple instances of the file to be recovered, the one most

recently saved shall be recovered, and an informational message that there are previous versions of the file that can be recovered shall be written. The editor shall behave as if the contents of the edit buffer have already been modified.

Current file: Set as described for the edit command.

Current column: Set as described for the edit command.

Rewind

Synopsis:

rew[ind][!]

If no '!' is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error, un? less the file is successfully written as specified by the autowrite op? tion.

If the argument list is empty, it shall be an error.

The current argument list reference and the current pathname shall be set to the first filename in the argument list.

Replace the contents of the edit buffer with the contents of the file named by the current pathname. If for any reason the contents of the file cannot be accessed, the edit buffer shall be empty.

This command shall be affected by the autowrite and writeany edit op? tions.

Current line: Set as described for the edit command.

Current column: Set as described for the edit command.

Set

Synopsis:

se[t][option[=[value]] ...][nooption ...][option? ...][all]

When no arguments are specified, write the value of the term edit op? tion and 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> characters. The '?' shall

be necessary only for Boolean valued options. Boolean options can be given values by the form set option to turn them on or set nooption to turn them off; string and numeric options can be assigned by the form set option=value. Any <blank> characters in strings can be included as is by preceding each <blank> with an escaping <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> characters.

See Edit Options in ex for details about specific options.

Current line: Unchanged.

Current column: Unchanged.

Shell

Synopsis:

sh[ell]

Invoke the program named in the shell edit option with the single argu? ment -i (interactive mode). Editing shall be resumed when the program exits.

Current line: Unchanged.

Current column: Unchanged.

Source

Synopsis:

so[urce] file

Read and execute ex commands from file. Lines in the file that are blank lines shall be ignored.

Current line: As specified for the individual ex commands.

Current column: As specified for the individual ex commands.

Substitute

Synopsis:

[2addr] s[ubstitute][/pattern/repl/[options][count][flags]]

[2addr] &[options][count][flags]]

[2addr] ~[options][count][flags]]

Replace the first instance of the pattern pattern by the string repl on each specified line. (See Regular Expressions in ex and Replacement

Strings in ex.) Any non-alphabetic, non-<blank> delimiter other than <backslash>, '|', <newline>, or double-quote can be used instead of '/'. <backslash> characters can be used to escape delimiters, <back? slash> characters, and other special characters.

The trailing delimiter can be omitted from pattern or from repl at the end of the command line. If both pattern and repl are not specified or are empty (for example, "//"), the last s command shall be repeated. If only pattern is not specified or is empty, the last regular expression used in the editor shall be used as the pattern. If only repl is not specified or is empty, the pattern shall be replaced by nothing. If the entire replacement pattern is '%', the last replacement pattern to an s command shall be used.

Entering a <carriage-return> in repl (which requires an escaping <back? slash> in ex mode and an escaping <control>?V in open or vi mode) shall split the line at that point, creating a new line in the edit buffer. The <carriage-return> shall be discarded.

If options includes the letter 'g' (global), all non-overlapping in? stances of the pattern in the line shall be replaced.

If options includes the letter 'c' (confirm), then before each substi? tution the line shall be written; the written line shall reflect all previous substitutions. On the following line, <space> characters shall be written beneath the characters from the line that are before the pattern to be replaced, and '^' characters written beneath the charac? ters included in the pattern to be replaced. The ex utility shall then wait for a response from the user. An affirmative response shall cause the substitution to be done, while any other input shall not make the substitution. An affirmative response shall consist of a line with the affirmative response (as defined by the current locale) at the begin? ning of the line. This line shall be subject to editing in the same way as the ex command line.

If interrupted (see the ASYNCHRONOUS EVENTS section), any modifications confirmed by the user shall be preserved in the edit buffer after the interrupt.

If the remembered search direction is not set, the s command shall set it to forward.

In the second Synopsis, the & command shall repeat the previous substi? tution, as if the & command were replaced by:

s/pattern/repl/

where pattern and repl are as specified in the previous s, &, or ~ com? mand.

In the third Synopsis, the ~ command shall repeat the previous substi? tution, as if the '~' were replaced by:

s/pattern/repl/

where pattern shall be the last regular expression specified to the ed? itor, and repl shall be from the previous substitution (including & and ~) command.

These commands shall be affected by the LC_MESSAGES environment vari? able.

Current line: Set to the last line in which a substitution occurred, or, unchanged if no substitution occurred.

Current column: Set to non-<blank>.

Suspend

Synopsis:

su[spend][!]

st[op][!]

Allow control to return to the invoking process; ex shall suspend it? self as if it had received the SIGTSTP signal. The suspension shall oc? cur only if job control is enabled in the invoking shell (see the de? scription of set -m).

These commands shall be affected by the autowrite and writeany edit op? tions.

The current susp character (see stty) shall be equivalent to the sus? pend command.

Tag

Synopsis:

ta[g][!] tagstring

The results are unspecified if the format of a tags file is not as specified by the ctags utility (see ctags) description.

The tag command shall search for tagstring in the tag files referred to by the tag edit option, in the order they are specified, until a refer? ence to tagstring is found. Files shall be searched from beginning to end. If no reference is found, it shall be an error and an error mes? sage to this effect shall be written. If the reference is not found, or if an error occurs while processing a file referred to in the tag edit option, it shall be an error, and an error message shall be written at the first occurrence of such an error.

Otherwise, if the tags file contained a pattern, the pattern shall be treated as a regular expression used in the editor; for example, for the purposes of the s command.

If the tagstring is in a file with a different name than the current pathname, set the current pathname to the name of that file, and re? place the contents of the edit buffer with the contents of that file.

In this case, if no '!' is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error, unless the file is successfully written as specified by the au? towrite option.

This command shall be affected by the autowrite, tag, taglength, and writeany edit options.

Current line: If the tags file contained a line number, set to that line number. If the line number is larger than the last line in the edit buffer, an error message shall be written and the current line shall be set as specified for the edit command.

If the tags file contained a pattern, set to the first occurrence of the pattern in the file. If no matching pattern is found, an error mes? sage shall be written and the current line shall be set as specified for the edit command.

Current column: If the tags file contained a line-number reference and that line-number was not larger than the last line in the edit buffer, or if the tags file contained a pattern and that pattern was found, set

to non-<blank>. Otherwise, set as specified for the edit command.

Unabbreviate

Synopsis:

una[bbrev] lhs

If lhs is not an entry in the current list of abbreviations (see Abbre?viate), it shall be an error. Otherwise, delete lhs from the list of abbreviations.

Current line: Unchanged.

Current column: Unchanged.

Undo

Synopsis:

u[ndo]

Reverse the changes made by the last command that modified the contents of the edit buffer, including undo. For this purpose, the global, v, open, and visual commands, and commands resulting from buffer execu?tions and mapped character expansions, are considered single commands.

If no action that can be undone preceded the undo command, it shall be an error.

If the undo command restores lines that were marked, the mark shall also be restored unless it was reset subsequent to the deletion of the lines.

Current line:

1. If lines are added or changed in the file, set to the first line added or changed.
2. Set to the line before the first line deleted, if it exists.
3. Set to 1 if the edit buffer is not empty.
4. Set to zero.

Current column: Set to non-<blank>.

Unmap

Synopsis:

unm[ap][!] lhs

If '!' is appended to the command name, and if lhs is not an entry in the list of text input mode map definitions, it shall be an error. Oth?

erwise, delete lhs from the list of text input mode map definitions.

If no '!' is appended to the command name, and if lhs is not an entry in the list of command mode map definitions, it shall be an error. Oth? erwise, delete lhs from the list of command mode map definitions.

Current line: Unchanged.

Current column: Unchanged.

Version

Synopsis:

ve[rsion]

Write a message containing version information for the editor. The for? mat of the message is unspecified.

Current line: Unchanged.

Current column: Unchanged.

Visual

Synopsis:

[1addr] vi[sual][type][count][flags]

If ex is currently in open or visual mode, the Synopsis and behavior of the visual command shall be the same as the edit command, as specified by Edit.

Otherwise, this command need not be supported on block-mode terminals or terminals with insufficient capabilities. If standard input, stan? dard output, or standard error are not terminal devices, the results are unspecified.

If count is specified, the value of the window edit option shall be set to count (as described in window). If the '^' type character was also specified, the window edit option shall be set before being used by the type character.

Enter visual mode. If type is not specified, it shall be as if a type of '+' was specified. The type shall cause the following effects:

+ Place the beginning of the specified line at the top of the dis? play.

- Place the end of the specified line at the bottom of the display.

. Place the beginning of the specified line in the middle of the

display.

^ If the specified line is less than or equal to the value of the window edit option, set the line to 1; otherwise, decrement the line by the value of the window edit option minus 1. Place the beginning of this line as close to the bottom of the displayed lines as possible, while still displaying the value of the window edit option number of lines.

Current line: Set to the specified line.

Current column: Set to non-<blank>.

Write

Synopsis:

[2addr] w[rite][!][>>][file]

[2addr] w[rite][!][file]

[2addr] wq[!][>>][file]

If no lines are specified, the lines shall default to the entire file.

The command wq shall be equivalent to a write command followed by a quit command; wq! shall be equivalent to write! followed by quit. In both cases, if the write command fails, the quit shall not be at? tempted.

If the command name is not followed by one or more <blank> characters, or file is not preceded by a '!' character, the write shall be to a file.

1. If the >> argument is specified, and the file already exists, the lines shall be appended to the file instead of replacing its con? tents. If the >> argument is specified, and the file does not al? ready exist, it is unspecified whether the write shall proceed as if the >> argument had not been specified or if the write shall fail.
2. If the readonly edit option is set (see readonly), the write shall fail.
3. If file is specified, and is not the current pathname, and the file exists, the write shall fail.
4. If file is not specified, the current pathname shall be used. If

there is no current pathname, the write command shall fail.

5. If the current pathname is used, and the current pathname has been changed by the file or read commands, and the file exists, the write shall fail. If the write is successful, subsequent writes shall not fail for this reason (unless the current pathname is changed again).

6. If the whole edit buffer is not being written, and the file to be written exists, the write shall fail.

For rules 1., 2., 3., and 5., the write can be forced by appending the character '!' to the command name.

For rules 2., 3., and 5., the write can be forced by setting the writeany edit option.

Additional, implementation-defined tests may cause the write to fail.

If the edit buffer is empty, a file without any contents shall be writ? ten.

An informational message shall be written noting the number of lines and bytes written.

Otherwise, if the command is followed by one or more <blank> charac? ters, and the file is preceded by '!', the rest of the line after the '!' shall have '%', '#', and '!' characters expanded as described in Command Line Parsing in ex.

The ex utility shall then pass two arguments to the program named by the shell edit option; the first shall be -c and the second shall be the expanded arguments to the write command as a single argument. The specified lines shall be written to the standard input of the command.

The standard error and standard output of the program, if any, shall be written as described for the print command. If the last character in that output is not a <newline>, a <newline> shall be written at the end of the output.

The special meaning of the '!' following the write command can be overridden by escaping it with a <backslash> character.

Current line: Unchanged.

Current column: Unchanged.

Write and Exit

Synopsis:

[2addr] x[it][!][file]

If the edit buffer has not been modified since the last complete write, xit shall be equivalent to the quit command, or if a '!' is appended to the command name, to quit!.

Otherwise, xit shall be equivalent to the wq command, or if a '!' is appended to the command name, to wq!.

Current line: Unchanged.

Current column: Unchanged.

Yank

Synopsis:

[2addr] ya[nk][buffer][count]

Copy the specified lines to the specified buffer (by default, the un? named buffer), which shall become a line-mode buffer.

Current line: Unchanged.

Current column: Unchanged.

Adjust Window

Synopsis:

[1addr] z[!][type ...][count][flags]

If no line is specified, the current line shall be the default; if type is omitted as well, the current line value shall first be incremented by 1. If incrementing the current line would cause it to be greater than the last line in the edit buffer, it shall be an error.

If there are <blank> characters between the type argument and the pre? ceding z command name or optional '!' character, it shall be an error.

If count is specified, the value of the window edit option shall be set to count (as described in window). If count is omitted, it shall de? fault to 2 times the value of the scroll edit option, or if ! was specified, the number of lines in the display minus 1.

If type is omitted, then count lines starting with the specified line shall be written. Otherwise, count lines starting with the line speci? fied by the type argument shall be written.

The type argument shall change the lines to be written. The possible values of type are as follows:

- The specified line shall be decremented by the following value:

  (((number of '-' characters) x count) -1)

  If the calculation would result in a number less than 1, it shall be an error. Write lines from the edit buffer, starting at the new value of line, until count lines or the last line in the edit buffer has been written.

+ The specified line shall be incremented by the following value:

  (((number of '+' characters) -1) x count) +1

  If the calculation would result in a number greater than the last line in the edit buffer, it shall be an error. Write lines from the edit buffer, starting at the new value of line, until count lines or the last line in the edit buffer has been written.

=,. If more than a single '.' or '=' is specified, it shall be an error. The following steps shall be taken:

1. If count is zero, nothing shall be written.
2. Write as many of the N lines before the current line in the edit buffer as exist. If count or '!' was specified, N shall be:

   (count -1) /2

   Otherwise, N shall be:

   (count -3) /2

   If N is a number less than 3, no lines shall be written.
3. If '=' was specified as the type character, write a line con? sisting of the smaller of the number of columns in the dis? play divided by two, or 40 '-' characters.
4. Write the current line.
5. Repeat step 3.
6. Write as many of the N lines after the current line in the edit buffer as exist. N shall be defined as in step 2. If N is a number less than 3, no lines shall be written. If count is less than 3, no lines shall be written.

^ The specified line shall be decremented by the following value:

(((number of '^' characters) +1) x count) -1

If the calculation would result in a number less than 1, it shall be an error. Write lines from the edit buffer, starting at the new value of line, until count lines or the last line in the edit buffer has been written.

Current line: Set to the last line written, unless the type is =, in which case, set to the specified line.

Current column: Set to non-<blank>.

Escape

Synopsis:

! command

[addr]! command

The contents of the line after the '!' shall have '%', '#', and '!' characters expanded as described in Command Line Parsing in ex. If the expansion causes the text of the line to change, it shall be redis? played, preceded by a single '!' character.

The ex utility shall execute the program named by the shell edit op? tion. It shall pass two arguments to the program; the first shall be -c, and the second shall be the expanded arguments to the ! command as a single argument.

If no lines are specified, the standard input, standard output, and standard error of the program shall be set to the standard input, stan? dard output, and standard error of the ex program when it was invoked.

In addition, a warning message shall be written if the edit buffer has been modified since the last complete write, and the warn edit option is set.

If lines are specified, they shall be passed to the program as standard input, and the standard output and standard error of the program shall replace those lines in the edit buffer. Each line in the program output (as delimited by <newline> characters or the end of the output if it is not immediately preceded by a <newline>), shall be a separate line in the edit buffer. Any occurrences of <carriage-return> and <newline>

pairs in the output shall be treated as single <newline> characters.

The specified lines shall be copied into the unnamed buffer before they are replaced, and the unnamed buffer shall become a line-mode buffer.

If in ex mode, a single '!' character shall be written when the pro? gram completes.

This command shall be affected by the shell and warn edit options. If no lines are specified, this command shall be affected by the autowrite and writeany edit options. If lines are specified, this command shall be affected by the autoprint edit option.

Current line:

1. If no lines are specified, unchanged.
2. Otherwise, set to the last line read in, if any lines are read in.
3. Otherwise, set to the line before the first line of the lines spec? ified, if that line exists.
4. Otherwise, set to the first line of the edit buffer if the edit buffer is not empty.
5. Otherwise, set to zero.

Current column: If no lines are specified, unchanged. Otherwise, set to non-<blank>.

Shift Left

Synopsis:

[2addr] <[< ...][count][flags]

Shift the specified lines to the start of the line; the number of col? umn positions to be shifted shall be the number of command characters times the value of the shiftwidth edit option. Only leading <blank> characters shall be deleted or changed into other <blank> characters in shifting; other characters shall not be affected.

Lines to be shifted shall be copied into the unnamed buffer, which shall become a line-mode buffer.

This command shall be affected by the autoprint edit option.

Current line: Set to the last line in the lines specified.

Current column: Set to non-<blank>.

Shift Right

Synopsis:

[2addr] >[> ...][count][flags]

Shift the specified lines away from the start of the line; the number of column positions to be shifted shall be the number of command characters times the value of the shiftwidth edit option. The shift shall be accomplished by adding <blank> characters as a prefix to the line or changing leading <blank> characters into other <blank> characters.

Empty lines shall not be changed.

Lines to be shifted shall be copied into the unnamed buffer, which shall become a line-mode buffer.

This command shall be affected by the autoprint edit option.

Current line: Set to the last line in the lines specified.

Current column: Set to non-<blank>.

<control>?D

Synopsis:

<control>-D

Write the next n lines, where n is the minimum of the values of the scroll edit option and the number of lines after the current line in the edit buffer. If the current line is the last line of the edit buffer it shall be an error.

Current line: Set to the last line written.

Current column: Set to non-<blank>.

Write Line Number

Synopsis:

[1addr] = [flags]

If line is not specified, it shall default to the last line in the edit buffer. Write the line number of the specified line.

Current line: Unchanged.

Current column: Unchanged.

Execute

Synopsis:

[2addr] @ buffer

[2addr] * buffer

If no buffer is specified or is specified as '@' or '*', the last buf?
fer executed shall be used. If no previous buffer has been executed, it
shall be an error.

For each line specified by the addresses, set the current line ('.')
to the specified line, and execute the contents of the named buffer (as
they were at the time the @ command was executed) as ex commands. For
each line of a line-mode buffer, and all but the last line of a charac?
ter-mode buffer, the ex command parser shall behave as if the line was
terminated by a <newline>.

If an error occurs during this process, or a line specified by the ad?
dresses does not exist when the current line would be set to it, or
more than a single line was specified by the addresses, and the con?
tents of the edit buffer are replaced (for example, by the ex :edit
command) an error message shall be written, and no more commands re?
sulting from the execution of this command shall be processed.

Current line: As specified for the individual ex commands.

Current column: As specified for the individual ex commands.

Regular Expressions in ex

The ex utility shall support regular expressions that are a superset of
the basic regular expressions described in the Base Definitions volume
of POSIX.1?2017, Section 9.3, Basic Regular Expressions. A null regu?
lar expression ("//") shall be equivalent to the last regular expres?
sion encountered.

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

The following constructs can be used to enhance the basic regular ex?
pressions:

\< Match the beginning of a word. (See the definition of word at
the beginning of Command Descriptions in ex.)

\> 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. The <back? slash> shall be discarded.

When the editor option magic is not set, the only characters with spe? cial meanings shall be '^' at the beginning of a pattern, '$' at the end of a pattern, and <backslash>. The characters '.', '*', '[', and '~' shall be treated as ordinary characters unless preceded by a <back? slash>; when preceded by a <backslash> they shall regain their special meaning, or in the case of <backslash>, be handled as a single <back? slash>. <backslash> characters used to escape other characters shall be discarded.

Replacement Strings in ex

The character '&' ('\&' if the editor option magic is not set) in the replacement string shall stand for the text matched by the pattern to be replaced. The character '~' ('\~' if magic is not set) shall be re? placed 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 corresponding back-reference expression. If the corre? sponding back-reference expression does not match, then the characters '\n' shall be replaced by the empty string.

The strings '\l', '\u', '\L', and '\U' can be used to modify the case of elements in the replacement string (using the '\&' or "\"digit) no? tation. The string '\l' ('\u') shall cause the character that follows to be converted to lowercase (uppercase). The string '\L' ('\U') shall cause all characters subsequent to it to be converted to lowercase (up? percase) as they are inserted by the substitution until the string '\e' or '\E', or the end of the replacement string, is encountered.

Otherwise, any character following a <backslash> shall be treated as that literal character, and the escaping <backslash> shall be dis? carded.

An example of case conversion with the s command is as follows:

```
: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.

Edit Options in ex

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

Options are Boolean unless otherwise specified.

autoindent, ai

[Default unset]

If autoindent is set, each line in input mode shall be indented (using first as many <tab> characters as possible, as determined by the editor option tabstop, and then using <space> characters) to align with an? other line, as follows:

1. If in open or visual mode and the text input is part of a line-ori? ented command (see the EXTENDED DESCRIPTION in vi), align to the first column.
2. Otherwise, if in open or visual mode, indentation for each line shall be set as follows:
   a. If a line was previously inserted as part of this command, it shall be set to the indentation of the last inserted line by default, or as otherwise specified for the <control>?D charac? ter in Input Mode Commands in vi.
   b. Otherwise, it shall be set to the indentation of the previous current line, if any; otherwise, to the first column.
3. For the ex a, i, and c commands, indentation for each line shall be set as follows:
   a. If a line was previously inserted as part of this command, it shall be set to the indentation of the last inserted line by default, or as otherwise specified for the eof character in Scroll.
   b. Otherwise, if the command is the ex a command, it shall be set to the line appended after, if any; otherwise to the first col?

umn.

c. Otherwise, if the command is the ex i command, it shall be set to the line inserted before, if any; otherwise to the first column.

d. Otherwise, if the command is the ex c command, it shall be set to the indentation of the line replaced.

autoprint, ap

[Default set]

If autoprint is set, the current line shall be written after each ex command that modifies the contents of the current edit buffer, and af? ter each tag command for which the tag search pattern was found or tag line number was valid, unless:

1. The command was executed while in open or visual mode.
2. The command was executed as part of a global or v command or @ buf? fer execution.
3. The command was the form of the read command that reads a file into the edit buffer.
4. The command was the append, change, or insert command.
5. The command was not terminated by a <newline>.
6. The current line shall be written by a flag specified to the com? mand; for example, delete # shall write the current line as speci? fied for the flag modifier to the delete command, and not as speci? fied by the autoprint edit option.

autowrite, aw

[Default unset]

If autowrite is set, and the edit buffer has been modified since it was last completely written to any file, the contents of the edit buffer shall be written as if the ex write command had been specified without arguments, before each command affected by the autowrite edit option is executed. Appending the character '!' to the command name of any of the ex commands except '!' shall prevent the write. If the write fails, it shall be an error and the command shall not be executed.

beautify, bf

[Default unset]

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

directory, dir

[Default implementation-defined]

The value of this option specifies the directory in which the editor buffer is to be placed. If this directory is not writable by the user, the editor shall quit.

edcompatible, ed

[Default unset]

Causes the presence of g and c suffixes on substitute commands to be remembered, and toggled by repeating the suffixes.

errorbells, eb

[Default unset]

If the editor is in ex mode, and the terminal does not support a stand? out mode (such as inverse video), and errorbells is set, error messages shall be preceded by alerting the terminal.

exrc

[Default unset]

If exrc is set, ex shall access any .exrc file in the current direc? tory, as described in Initialization in ex and vi. If exrc is not set, ex shall ignore any .exrc file in the current directory during initial? ization, unless the current directory is that named by the HOME envi? ronment variable.

ignorecase, ic

[Default unset]

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

The ignorecase edit option shall affect all remembered regular expres? sions; for example, unsetting the ignorecase edit option shall cause a subsequent vi n command to search for the last basic regular expression

in a case-sensitive fashion.

list

[Default unset]

If list is set, edit buffer lines written while in ex command mode shall be written as specified for the print command with the l flag specified. In open or visual mode, each edit buffer line shall be dis? played as specified for the ex print command with the l flag specified.

In open or visual text input mode, when the cursor does not rest on any character in the line, it shall rest on the '$' marking the end of the line.

magic

[Default set]

If magic is set, modify the interpretation of characters in regular ex? pressions and substitution replacement strings (see Regular Expressions in ex and Replacement Strings in ex).

mesg

[Default set]

If mesg is set, the permission for others to use the write or talk com? mands to write to the terminal shall be turned on while in open or vis? ual mode. The shell-level command mesg n shall take precedence over any setting of the ex mesg option; that is, if mesg y was issued before the editor started (or in a shell escape), such as:

```
:!mesg y
```

the mesg option in ex shall suppress incoming messages, but the mesg option shall not enable incoming messages if mesg n was issued.

number, nu

[Default unset]

If number is set, edit buffer lines written while in ex command mode shall be written with line numbers, in the format specified by the print command with the # flag specified. In ex text input mode, each line shall be preceded by the line number it will have in the file.

In open or visual mode, each edit buffer line shall be displayed with a preceding line number, in the format specified by the ex print command

with the # flag specified. This line number shall not be considered part of the line for the purposes of evaluating the current column; that is, column position 1 shall be the first column position after the format specified by the print command.

paragraphs, para

[Default in the POSIX locale IPLPPPQPP LIpplpipbp]

The paragraphs edit option shall define additional paragraph boundaries for the open and visual mode commands. The paragraphs edit option can be set to a character string consisting of zero or more character pairs. It shall be an error to set it to an odd number of characters.

prompt

[Default set]

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

readonly

[Default see text]

If the readonly edit option is set, read-only mode shall be enabled (see Write). The readonly edit option shall be initialized to set if either of the following conditions are true:

* The command-line option -R was specified.
* Performing actions equivalent to the access() function called with the following arguments indicates that the file lacks write permis? sion:
  1. The current pathname is used as the path argument.
  2. The constant W_OK is used as the amode argument.

The readonly edit option may be initialized to set for other, implemen? tation-defined reasons. The readonly edit option shall not be initial? ized to unset based on any special privileges of the user or process. The readonly edit option shall be reinitialized each time that the con? tents of the edit buffer are replaced (for example, by an edit or next command) unless the user has explicitly set it, in which case it shall remain set until the user explicitly unsets it. Once unset, it shall again be reinitialized each time that the contents of the edit buffer

are replaced.

redraw

[Default unset]

The editor simulates an intelligent terminal on a dumb terminal. (Since this is likely to require a large amount of output to the termi? nal, it is useful only at high transmission speeds.)

remap

[Default set]

If remap is set, map translation shall allow for maps defined in terms of other maps; translation shall continue until a final product is ob? tained. If unset, only a one-step translation shall be done.

report

[Default 5]

The value of this report edit option specifies what number of lines be? ing added, copied, deleted, or modified in the edit buffer will cause an informational message to be written to the user. The following con? ditions shall cause an informational message. The message shall contain the number of lines added, copied, deleted, or modified, but is other? wise unspecified.

* An ex or vi editor command, other than open, undo, or visual, that modifies at least the value of the report edit option number of lines, and which is not part of an ex global or v command, or ex or vi buffer execution, shall cause an informational message to be written.
* An ex yank or vi y or Y command, that copies at least the value of the report edit option plus 1 number of lines, and which is not part of an ex global or v command, or ex or vi buffer execution, shall cause an informational message to be written.
* An ex global, v, open, undo, or visual command or ex or vi buffer execution, that adds or deletes a total of at least the value of the report edit option number of lines, and which is not part of an ex global or v command, or ex or vi buffer execution, shall cause an informational message to be written. (For example, if 3 lines

were added and 8 lines deleted during an ex visual command, 5 would be the number compared against the report edit option after the command completed.)

scroll, scr

[Default (number of lines in the display -1)/2]

The value of the scroll edit option shall determine the number of lines scrolled by the ex <control>?D and z commands. For the vi <control>?D and <control>?U commands, it shall be the initial number of lines to scroll when no previous <control>?D or <control>?U command has been ex?ecuted.

sections

[Default in the POSIX locale NHSHH HUnhsh]

The sections edit option shall define additional section boundaries for the open and visual mode commands. The sections edit option can be set to a character string consisting of zero or more character pairs; it shall be an error to set it to an odd number of characters.

shell, sh

[Default from the environment variable SHELL]

The value of this option shall be a string. The default shall be taken from the SHELL environment variable. If the SHELL environment variable is null or empty, the sh (see sh) utility shall be the default.

shiftwidth, sw

[Default 8]

The value of this option shall give the width in columns of an indenta?tion level used during autoindentation and by the shift commands (< and >).

showmatch, sm

[Default unset]

The functionality described for the showmatch edit option need not be supported on block-mode terminals or terminals with insufficient capa?bilities.

If showmatch is set, in open or visual mode, when a ')' or '}' is typed, if the matching '(' or '{' is currently visible on the display,

the matching '(' or '{' shall be flagged moving the cursor to its loca?
tion for an unspecified amount of time.

showmode

[Default unset]

If showmode is set, in open or visual mode, the current mode that the editor is in shall be displayed on the last line of the display. Com?
mand mode and text input mode shall be differentiated; other unspeci?
fied modes and implementation-defined information may be displayed.

slowopen

[Default unset]

If slowopen is set during open and visual text input modes, the editor shall not update portions of the display other than those display line columns that display the characters entered by the user (see Input Mode Commands in vi).

tabstop, ts

[Default 8]

The value of this edit option shall specify the column boundary used by a <tab> in the display (see autoprint, ap and Input Mode Commands in vi).

taglength, tl

[Default zero]

The value of this edit option shall specify the maximum number of char?
acters that are considered significant in the user-specified tag name and in the tag name from the tags file. If the value is zero, all char?
acters in both tag names shall be significant.

tags

[Default see text]

The value of this edit option shall be a string of <blank>-delimited pathnames of files used by the tag command. The default value is un?
specified.

term

[Default from the environment variable TERM]

The value of this edit option shall be a string. The default shall be

taken from the TERM variable in the environment. If the TERM environ?
ment variable is empty or null, the default is unspecified. The editor
shall use the value of this edit option to determine the type of the
display device.

The results are unspecified if the user changes the value of the term
edit option after editor initialization.

terse

[Default unset]

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.

warn

[Default set]

If warn is set, and the contents of the edit buffer have been modified
since they were last completely written, the editor shall write a warn?
ing message before certain ! commands (see Escape).

window

[Default see text]

A value used in open and visual mode, by the <control>?B and <con?
trol>?F commands, and, in visual mode, to specify the number of lines
displayed when the screen is repainted.

If the -w command-line option is not specified, the default value shall
be set to the value of the LINES environment variable. If the LINES en?
vironment variable is empty or null, the default shall be the number of
lines in the display minus 1.

Setting the window edit option to zero or to a value greater than the
number of lines in the display minus 1 (either explicitly or based on
the -w option or the LINES environment variable) shall cause the window
edit option to be set to the number of lines in the display minus 1.

The baud rate of the terminal line may change the default in an imple?
mentation-defined manner.

wrapmargin, wm

[Default 0]

If the value of this edit option is zero, it shall have no effect.

If not in the POSIX locale, the effect of this edit option is implemen?

tation-defined.

Otherwise, it shall specify a number of columns from the ending margin

of the terminal.

During open and visual text input modes, for each character for which

any part of the character is displayed in a column that is less than

wrapmargin columns from the ending margin of the display line, the edi?

tor shall behave as follows:

1. If the character triggering this event is a <blank>, it, and all

   immediately preceding <blank> characters on the current line en?

   tered during the execution of the current text input command, shall

   be discarded, and the editor shall behave as if the user had en?

   tered a single <newline> instead. In addition, if the next user-en?

   tered character is a <space>, it shall be discarded as well.

2. Otherwise, if there are one or more <blank> characters on the cur?

   rent line immediately preceding the last group of inserted

   non-<blank> characters which was entered during the execution of

   the current text input command, the <blank> characters shall be re?

   placed as if the user had entered a single <newline> instead.

If the autoindent edit option is set, and the events described in 1. or

2. are performed, any <blank> characters at or after the cursor in the

current line shall be discarded.

The ending margin shall be determined by the system or overridden by

the user, as described for COLUMNS in the ENVIRONMENT VARIABLES section

and the Base Definitions volume of POSIX.1?2017, Chapter 8, Environment

Variables.

wrapscan, ws

[Default set]

If wrapscan is set, searches (the ex / or ? addresses, or open and

visual mode /, ?, N, and n commands) shall wrap around the beginning or

end of the edit buffer; when unset, searches shall stop at the begin?

ning or end of the edit buffer.

writeany, wa

[Default unset]

If writeany is set, some of the checks performed when executing the ex write commands shall be inhibited, as described in editor option au? towrite.

EXIT STATUS

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

When any error is encountered and the standard input is not a terminal device file, ex shall not write the file or return to command or text input mode, and shall terminate with a non-zero exit status.

Otherwise, when an unrecoverable error is encountered, it shall be equivalent to a SIGHUP asynchronous event.

Otherwise, when an error is encountered, the editor shall behave as specified in Command Line Parsing in ex.

The following sections are informative.

APPLICATION USAGE

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 ex? ample, because they expect only one file and unspecified results occur.

EXAMPLES

None.

RATIONALE

The ex/vi specification is based on the historical practice found in the 4 BSD and System V implementations of ex and vi.

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

It is recognized that ex visual mode and related features would be dif? ficult, if not impossible, to implement satisfactorily on a block-mode terminal, or a terminal without any form of cursor addressing; 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.

Options

The -c replacement for +command was inspired by the -e option of sed. Historically, all such commands (see edit and next as well) were exe? cuted from the last line of the edit buffer. This meant, for example, that "+/pattern" would fail unless the wrapscan option was set. POSIX.1?2008 requires conformance to historical practice. The +command option is no longer specified by POSIX.1?2008 but may be present in some implementations. Historically, some implementations restricted the ex commands that could be listed as part of the command line arguments. For consistency, POSIX.1?2008 does not permit these restrictions.

In historical implementations of the editor, the -R option (and the readonly edit option) only prevented overwriting of files; appending to files was still permitted, mapping loosely into the csh noclobber vari? able. Some implementations, however, have not followed this semantic, and readonly does not permit appending either. POSIX.1?2008 follows the latter practice, believing that it is a more obvious and intuitive meaning of readonly.

The -s option suppresses all interactive user feedback and is useful for editing scripts in batch jobs. The list of specific effects is his? torical practice. The terminal type ``incapable of supporting open and visual modes'' has historically been named ``dumb''.

The -t option was required because the ctags utility appears in POSIX.1?2008 and the option is available in all historical implementa? tions of ex.

Historically, the ex and vi utilities accepted a -x option, which did encryption based on the algorithm found in the historical crypt util?

ity. 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. In addition, it did not historically provide the level of security that users might expect.

Standard Input

An end-of-file condition is not equivalent to an end-of-file character. A common end-of-file character, <control>?D, is historically an ex command.

There was no maximum line length in historical implementations of ex. Specifically, as it was parsed in chunks, the addresses had a different maximum length than the filenames. Further, the maximum line buffer size was declared as BUFSIZ, which was different lengths on different systems. This version selected the value of {LINE_MAX} to impose a reasonable restriction on portable usage of ex and to aid test suite writers in their development of realistic tests that exercise this limit.

Input Files

It was an explicit decision by the standard developers that a <newline> be added to any file lacking one. It was believed that this feature of ex and vi was relied on by users in order to make text files lacking a trailing <newline> more portable. It is recognized that this will require a user-specified option or extension for implementations that permit ex and vi to edit files of type other than text if such files are not otherwise identified by the system. It was agreed that the ability to edit files of arbitrary type can be useful, but it was not considered necessary to mandate that an ex or vi implementation be required to handle files other than text files.

The paragraph in the INPUT FILES section, ``By default, ...'', is intended to close a long-standing security problem in ex and vi; that of the ``modeline'' or ``modelines'' edit option. This feature allows any line in the first or last five lines of the file containing the strings "ex:" or "vi:" (and, apparently, "ei:" or "vx:") 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 the editor when replying to a mail message in which a line such as:

ex:! rm -rf :

appeared in the signature lines. The standard developers believed strongly that an editor should not by default interpret any lines of a file. Vendors are strongly urged to delete this feature from their im? plementations of ex and vi.

Asynchronous Events

The intention of the phrase ``complete write'' is that the entire edit buffer be written to stable storage. The note regarding temporary files is intended for implementations that use temporary files to back edit buffers unnamed by the user.

Historically, SIGQUIT was ignored by ex, but was the equivalent of the Q command in visual mode; that is, it exited visual mode and entered ex mode. POSIX.1?2008 permits, but does not require, this behavior. His? torically, SIGINT was often used by vi users to terminate text input mode (<control>?C is often easier to enter than <ESC>). Some implemen? tations of vi alerted the terminal on this event, and some did not. POSIX.1?2008 requires that SIGINT behave identically to <ESC>, and that the terminal not be alerted.

Historically, suspending the ex editor during text input mode was simi? lar to SIGINT, as completed lines were retained, but any partial line discarded, and the editor returned to command mode. POSIX.1?2008 is silent on this issue; implementations are encouraged to follow histori? cal practice, where possible.

Historically, the vi editor did not treat SIGTSTP as an asynchronous event, and it was therefore impossible to suspend the editor in visual text input mode. There are two major reasons for this. The first is that SIGTSTP is a broadcast signal on UNIX systems, and the chain of events where the shell execs an application that then execs vi usually caused confusion for the terminal state if SIGTSTP was delivered to the process group in the default manner. The second was that most implemen?

tations of the UNIX curses package did not handle SIGTSTP safely, and the receipt of SIGTSTP at the wrong time would cause them to crash. POSIX.1?2008 is silent on this issue; implementations are encouraged to treat suspension as an asynchronous event if possible.

Historically, modifications to the edit buffer made before SIGINT in? terrupted an operation were retained; that is, anywhere from zero to all of the lines to be modified might have been modified by the time the SIGINT arrived. These changes were not discarded by the arrival of SIGINT. POSIX.1?2008 permits this behavior, noting that the undo com? mand is required to be able to undo these partially completed commands.

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

Standard Error

For ex/vi, diagnostic messages are those messages reported as a result of a failed attempt to invoke ex or vi, such as invalid options or in? sufficient resources, or an abnormal termination condition. Diagnostic messages should not be confused with the error messages generated by inappropriate or illegal user commands.

Initialization in ex and vi

If an ex command (other than cd, chdir, or source) has a filename argu? ment, one or both of the alternate and current pathnames will be set. Informally, they are set as follows:

1. If the ex command is one that replaces the contents of the edit buffer, and it succeeds, the current pathname will be set to the filename argument (the first filename argument in the case of the next command) and the alternate pathname will be set to the previ? ous current pathname, if there was one.
2. In the case of the file read/write forms of the read and write com? mands, if there is no current pathname, the current pathname will be set to the filename argument.
3. Otherwise, the alternate pathname will be set to the filename argu? ment.

For example, :edit foo and :recover foo, when successful, set the cur? rent pathname, and, if there was a previous current pathname, the al? ternate pathname. The commands :write, !command, and :edit set neither the current or alternate pathnames. If the :edit foo command were to fail for some reason, the alternate pathname would be set. The read and write commands set the alternate pathname to their file argument, un? less the current pathname is not set, in which case they set the cur? rent pathname to their file arguments. The alternate pathname was not historically set by the :source command. POSIX.1?2008 requires confor? mance to historical practice. Implementations adding commands that take filenames as arguments are encouraged to set the alternate path? name as described here.

Historically, ex and vi read the .exrc file in the $HOME directory twice, if the editor was executed in the $HOME directory. POSIX.1?2008 prohibits this behavior.

Historically, the 4 BSD ex and vi read the $HOME and local .exrc files if they were owned by the real ID of the user, or the sourceany option was set, regardless of other considerations. This was a security prob? lem because it is possible to put normal UNIX system commands inside a .exrc file. POSIX.1?2008 does not specify the sourceany option, and historical implementations are encouraged to delete it.

The .exrc files must be owned by the real ID of the user, and not writable by anyone other than the owner. The appropriate privileges ex? ception is intended to permit users to acquire special privileges, but continue to use the .exrc files in their home directories.

System V Release 3.2 and later vi implementations added the option [no]exrc. The behavior is that local .exrc files are read-only if the exrc option is set. The default for the exrc option was off, so by de? fault, local .exrc files were not read. The problem this was intended to solve was that System V permitted users to give away files, so there is no possible ownership or writeability test to ensure that the file is safe. This is still a security problem on systems where users can give away files, but there is nothing additional that POSIX.1?2008 can

do. The implementation-defined exception is intended to permit groups to have local .exrc files that are shared by users, by creating pseudo-users to own the shared files.

POSIX.1?2008 does not mention system-wide ex and vi start-up files. While they exist in several implementations of ex and vi, they are not present in any implementations considered historical practice by POSIX.1?2008. Implementations that have such files should use them only if they are owned by the real user ID or an appropriate user (for exam? ple, root on UNIX systems) and if they are not writable by any user other than their owner. System-wide start-up files should be read be? fore the EXINIT variable, $HOME/.exrc, or local .exrc files are evalu? ated.

Historically, any ex command could be entered in the EXINIT variable or the .exrc file, although ones requiring that the edit buffer already contain lines of text generally caused historical implementations of the editor to drop core. POSIX.1?2008 requires that any ex command be permitted in the EXINIT variable and .exrc files, for simplicity of specification and consistency, although many of them will obviously fail under many circumstances.

The initialization of the contents of the edit buffer uses the phrase ``the effect shall be'' with regard to various ex commands. The intent of this phrase is that edit buffer contents loaded during the initial? ization phase not be lost; that is, loading the edit buffer should fail if the .exrc file read in the contents of a file and did not subse? quently write the edit buffer. An additional intent of this phrase is to specify that the initial current line and column is set as specified for the individual ex commands.

Historically, the -t option behaved as if the tag search were a +com? mand; that is, it was executed from the last line of the file specified by the tag. This resulted in the search failing if the pattern was a forward search pattern and the wrapscan edit option was not set. POSIX.1?2008 does not permit this behavior, requiring that the search for the tag pattern be performed on the entire file, and, if not found,

that the current line be set to a more reasonable location in the file.

Historically, the empty edit buffer presented for editing when a file was not specified by the user was unnamed. This is permitted by POSIX.1?2008; however, implementations are encouraged to provide users a temporary filename for this buffer because it permits them the use of ex commands that use the current pathname during temporary edit ses? sions.

Historically, the file specified using the -t option was not part of the current argument list. This practice is permitted by POSIX.1?2008; however, implementations are encouraged to include its name in the cur? rent argument list for consistency.

Historically, the -c command was generally not executed until a file that already exists was edited. POSIX.1?2008 requires conformance to this historical practice. Commands that could cause the -c command to be executed include the ex commands edit, next, recover, rewind, and tag, and the vi commands <control>?^ and <control>?]. Historically, reading a file into an edit buffer did not cause the -c command to be executed (even though it might set the current pathname) with the ex? ception that it did cause the -c command to be executed if: the editor was in ex mode, the edit buffer had no current pathname, the edit buf? fer was empty, and no read commands had yet been attempted. For consis? tency and simplicity of specification, POSIX.1?2008 does not permit this behavior.

Historically, the -r option was the same as a normal edit session if there was no recovery information available for the file. This allowed users to enter:

```
vi -r *.c
```

and recover whatever files were recoverable. In some implementations, recovery was attempted only on the first file named, and the file was not entered into the argument list; in others, recovery was attempted for each file named. In addition, some historical implementations ig? nored -r if -t was specified or did not support command line file argu? ments with the -t option. For consistency and simplicity of specifica?

tion, POSIX.1?2008 disallows these special cases, and requires that re? covery be attempted the first time each file is edited.

Historically, vi initialized the ` and ' marks, but ex did not. This meant that if the first command in ex mode was visual or if an ex com? mand was executed first (for example, vi +10 file), vi was entered without the marks being initialized. Because the standard developers believed the marks to be generally useful, and for consistency and sim? plicity of specification, POSIX.1?2008 requires that they always be initialized if in open or visual mode, or if in ex mode and the edit buffer is not empty. Not initializing it in ex mode if the edit buffer is empty is historical practice; however, it has always been possible to set (and use) marks in empty edit buffers in open and visual mode edit sessions.

Addressing

Historically, ex and vi accepted the additional addressing forms '\/' and '\?'. They were equivalent to "//" and "??", respectively. They are not required by POSIX.1?2008, mostly because nobody can remember whether they ever did anything different historically.

Historically, ex and vi permitted an address of zero for several com? mands, and permitted the % address in empty files for others. For con? sistency, POSIX.1?2008 requires support for the former in the few com? mands where it makes sense, and disallows it otherwise. In addition, because POSIX.1?2008 requires that % be logically equivalent to "1,$", it is also supported where it makes sense and disallowed otherwise.

Historically, the % address could not be followed by further addresses. For consistency and simplicity of specification, POSIX.1?2008 requires that additional addresses be supported.

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.

3 ---- 2 Line one (note intermediate negative address).

1 2 3 Line six.

Any number of addresses can be provided to commands taking addresses; for example, "1,2,3,4,5p" prints lines 4 and 5, because two is the greatest valid number of addresses accepted by the print command. This, in combination with the <semicolon> delimiter, permits users to create commands based on ordered patterns in the file. For example, the com? mand 3;/foo/;+2print will display the first line after line 3 that con? tains the pattern foo, plus the next two lines. Note that the address 3; must be evaluated before being discarded because the search origin for the /foo/ command depends on this.

Historically, values could be added to addresses by including them af? ter one or more <blank> characters; for example, 3 - 5p wrote the sev? enth line of the file, and /foo/ 5 was the same as /foo/+5. However, only absolute values could be added; for example, 5 /foo/ was an error. POSIX.1?2008 requires conformance to historical practice. Address off? sets are separately specified from addresses because they could histor? ically be provided to visual mode search commands.

Historically, any missing addresses defaulted to the current line. This was true for leading and trailing <comma>-delimited addresses, and for trailing <semicolon>-delimited addresses. For consistency, POSIX.1?2008 requires it for leading <semicolon> addresses as well.

Historically, ex and vi accepted the '^' character as both an address and as a flag offset for commands. In both cases it was identical to the '-' character. POSIX.1?2008 does not require or prohibit this be? havior.

Historically, the enhancements to basic regular expressions could be used in addressing; for example, '~', '\<', and '\>'. POSIX.1?2008 re? quires conformance to historical practice; that is, that regular ex? pression usage be consistent, and that regular expression enhancements be supported wherever regular expressions are used.

Command Line Parsing in ex

Historical ex command parsing was even more complex than that described here. POSIX.1?2008 requires the subset of the command parsing that the standard developers believed was documented and that users could rea?

sonably be expected to use in a portable fashion, and that was histori? cally consistent between implementations. (The discarded functionality is obscure, at best.) Historical implementations will require changes in order to comply with POSIX.1?2008; however, users are not expected to notice any of these changes. Most of the complexity in ex parsing is to handle three special termination cases:

1. The !, global, v, and the filter versions of the read and write commands are delimited by <newline> characters (they can contain <vertical-line> characters that are usually shell pipes).
2. The ex, edit, next, and visual in open and visual mode commands all take ex commands, optionally containing <vertical-line> characters, as their first arguments.
3. The s command takes a regular expression as its first argument, and uses the delimiting characters to delimit the command.

Historically, <vertical-line> characters in the +command argument of the ex, edit, next, vi, and visual commands, and in the pattern and re? placement parts of the s command, did not delimit the command, and in the filter cases for read and write, and the !, global, and v commands, they did not delimit the command at all. For example, the following commands are all valid:

```
:edit +25 | s/abc/ABC/ file.c
:s/ | /PIPE/
:read !spell % | columnate
:global/pattern/p | l
:s/a/b/ | s/c/d | set
```

Historically, empty or <blank> filled lines in .exrc files and sourced files (as well as EXINIT variables and ex command scripts) were treated as default commands; that is, print commands. POSIX.1?2008 specifically requires that they be ignored when encountered in .exrc and sourced files to eliminate a common source of new user error.

Historically, ex commands with multiple adjacent (or <blank>-separated) vertical lines were handled oddly when executed from ex mode. For exam? ple, the command ||| <carriage-return>, when the cursor was on line 1,

displayed lines 2, 3, and 5 of the file. In addition, the command | would only display the line after the next line, instead of the next two lines. The former worked more logically when executed from vi mode, and displayed lines 2, 3, and 4. POSIX.1?2008 requires the vi behavior; that is, a single default command and line number increment for each command separator, and trailing <newline> characters after <vertical-line> separators are discarded.

Historically, ex permitted a single extra <colon> as a leading command character; for example, :g/pattern/:p was a valid command. POSIX.1?2008 generalizes this to require that any number of leading <colon> charac?ters be stripped.

Historically, any prefix of the delete command could be followed with?out intervening <blank> characters by a flag character because in the command d p, p is interpreted as the buffer p. POSIX.1?2008 requires conformance to historical practice.

Historically, the k command could be followed by the mark name without intervening <blank> characters. POSIX.1?2008 requires conformance to historical practice.

Historically, the s command could be immediately followed by flag and option characters; for example, s/e/E/|s|sgc3p was a valid command. However, flag characters could not stand alone; for example, the com?mands sp and s l would fail, while the command sgp and s gl would suc?ceed. (Obviously, the '#' flag character was used as a delimiter char?acter if it followed the command.) Another issue was that option char?acters had to precede flag characters even when the command was fully specified; for example, the command s/e/E/pg would fail, while the com?mand s/e/E/gp would succeed. POSIX.1?2008 requires conformance to his?torical practice.

Historically, the first command name that had a prefix matching the in?put from the user was the executed command; for example, ve, ver, and vers all executed the version command. Commands were in a specific or?der, however, so that a matched append, not abbreviate. POSIX.1?2008 requires conformance to historical practice. The restriction on command

search order for implementations with extensions is to avoid the addi?
tion of commands such that the historical prefixes would fail to work
portably.

Historical implementations of ex and vi did not correctly handle multi?
ple ex commands, separated by <vertical-line> characters, that entered
or exited visual mode or the editor. Because implementations of vi ex?
ist that do not exhibit this failure mode, POSIX.1?2008 does not permit
it.

The requirement that alphabetic command names consist of all following
alphabetic characters up to the next non-alphabetic character means
that alphabetic command names must be separated from their arguments by
one or more non-alphabetic characters, normally a <blank> or '!' char?
acter, except as specified for the exceptions, the delete, k, and s
commands.

Historically, the repeated execution of the ex default print commands
(<control>?D, eof, <newline>, <carriage-return>) erased any prompting
character and displayed the next lines without scrolling the terminal;
that is, immediately below any previously displayed lines. This pro?
vided a cleaner presentation of the lines in the file for the user.
POSIX.1?2008 does not require this behavior because it may be impossi?
ble in some situations; however, implementations are strongly encour?
aged to provide this semantic if possible.

Historically, it was possible to change files in the middle of a com?
mand, and have the rest of the command executed in the new file; for
example:

```
:edit +25 file.c | s/abc/ABC/ | 1
```

was a valid command, and the substitution was attempted in the newly
edited file. POSIX.1?2008 requires conformance to historical practice.

The following commands are examples that exercise the ex parser:

```
echo 'foo | bar' > file1; echo 'foo/bar' > file2;
vi
:edit +1 | s/|/PIPE/ | w file1 | e file2 | 1 | s/\//SLASH/ | wq
```

Historically, there was no protection in editor implementations to

avoid ex global, v, @, or * commands changing edit buffers during exe? cution of their associated commands. Because this would almost invari? ably result in catastrophic failure of the editor, and implementations exist that do exhibit these problems, POSIX.1?2008 requires that chang? ing the edit buffer during a global or v command, or during a @ or * command for which there will be more than a single execution, be an er? ror. Implementations supporting multiple edit buffers simultaneously are strongly encouraged to apply the same semantics to switching be? tween buffers as well.

The ex command quoting required by POSIX.1?2008 is a superset of the quoting in historical implementations of the editor. For example, it was not historically possible to escape a <blank> in a filename; for example, :edit foo\\\ bar would report that too many filenames had been entered for the edit command, and there was no method of escaping a <blank> in the first argument of an edit, ex, next, or visual command at all. POSIX.1?2008 extends historical practice, requiring that quot? ing behavior be made consistent across all ex commands, except for the map, unmap, abbreviate, and unabbreviate commands, which historically used <control>?V instead of <backslash> characters for quoting. For those four commands, POSIX.1?2008 requires conformance to historical practice.

Backslash quoting in ex is non-intuitive. <backslash>-escapes are ig? nored unless they escape a special character; for example, when per? forming file argument expansion, the string "\\%" is equivalent to '\%', not "\<current pathname>". This can be confusing for users be? cause <backslash> is usually one of the characters that causes shell expansion to be performed, and therefore shell quoting rules must be taken into consideration. Generally, quoting characters are only con? sidered if they escape a special character, and a quoting character must be provided for each layer of parsing for which the character is special. As another example, only a single <backslash> is necessary for the '\l' sequence in substitute replacement patterns, because the char? acter 'l' is not special to any parsing layer above it.

<control>?V quoting in ex is slightly different from backslash quoting.

In the four commands where <control>?V quoting applies (abbreviate, un? abbreviate, map, and unmap), any character may be escaped by a <con? trol>?V whether it would have a special meaning or not. POSIX.1?2008 requires conformance to historical practice.

Historical implementations of the editor did not require delimiters within character classes to be escaped; for example, the command :s/[/]// on the string "xxx/yyy" would delete the '/' from the string. POSIX.1?2008 disallows this historical practice for consistency and be? cause it places a large burden on implementations by requiring that knowledge of regular expressions be built into the editor parser.

Historically, quoting <newline> characters in ex commands was handled inconsistently. In most cases, the <newline> character always termi? nated the command, regardless of any preceding escape character, be? cause <backslash> characters did not escape <newline> characters for most ex commands. However, some ex commands (for example, s, map, and abbreviation) permitted <newline> characters to be escaped (although in the case of map and abbreviation, <control>?V characters escaped them instead of <backslash> characters). This was true in not only the com? mand line, but also .exrc and sourced files. For example, the command:

```
map = foo<control-V><newline>bar
```

would succeed, although it was sometimes difficult to get the <con? trol>?V and the inserted <newline> passed to the ex parser. For consis? tency and simplicity of specification, POSIX.1?2008 requires that it be possible to escape <newline> characters in ex commands at all times, using <backslash> characters for most ex commands, and using <con? trol>?V characters for the map and abbreviation commands. For example, the command print<newline>list is required to be parsed as the single command print<newline>list. While this differs from historical prac? tice, POSIX.1?2008 developers believed it unlikely that any script or user depended on the historical behavior.

Historically, an error in a command specified using the -c option did not cause the rest of the -c commands to be discarded. POSIX.1?2008

disallows this for consistency with mapped keys, the @, global, source, and v commands, the EXINIT environment variable, and the .exrc files.

Input Editing in ex

One of the common uses of the historical ex editor is over slow network connections. Editors that run in canonical mode can require far less traffic to and from, and far less processing on, the host machine, as well as more easily supporting block-mode terminals. For these reasons, POSIX.1?2008 requires that ex be implemented using canonical mode input processing, as was done historically.

POSIX.1?2008 does not require the historical 4 BSD input editing char? acters ``word erase'' or ``literal next''. For this reason, it is un? specified how they are handled by ex, although they must have the re? quired effect. Implementations that resolve them after the line has been ended using a <newline> or <control>?M character, and implementa? tions that rely on the underlying system terminal support for this pro? cessing, are both conforming. Implementations are strongly urged to use the underlying system functionality, if at all possible, for com? patibility with other system text input interfaces.

Historically, when the eof character was used to decrement the autoin? dent level, the cursor moved to display the new end of the autoindent characters, but did not move the cursor to a new line, nor did it erase the <control>?D character from the line. POSIX.1?2008 does not specify that the cursor remain on the same line or that the rest of the line is erased; however, implementations are strongly encouraged to provide the best possible user interface; that is, the cursor should remain on the same line, and any <control>?D character on the line should be erased.

POSIX.1?2008 does not require the historical 4 BSD input editing char? acter ``reprint'', traditionally <control>?R, which redisplayed the current input from the user. For this reason, and because the function? ality cannot be implemented after the line has been terminated by the user, POSIX.1?2008 makes no requirements about this functionality. Im? plementations are strongly urged to make this historical functionality available, if possible.

Historically, <control>?Q did not perform a literal next function in ex, as it did in vi. POSIX.1?2008 requires conformance to historical practice to avoid breaking historical ex scripts and .exrc files.

eof

Whether the eof character immediately modifies the autoindent charac? ters in the prompt is left unspecified so that implementations can con? form in the presence of systems that do not support this functionality. Implementations are encouraged to modify the line and redisplay it im? mediately, if possible.

The specification of the handling of the eof character differs from historical practice only in that eof characters are not discarded if they follow normal characters in the text input. Historically, they were always discarded.

Command Descriptions in ex

Historically, several commands (for example, global, v, visual, s, write, wq, yank, !, <, >, &, and ~) were executable in empty files (that is, the default address(es) were 0), or permitted explicit ad? dresses of 0 (for example, 0 was a valid address, or 0,0 was a valid range). Addresses of 0, or command execution in an empty file, make sense only for commands that add new text to the edit buffer or write commands (because users may wish to write empty files). POSIX.1?2008 requires this behavior for such commands and disallows it otherwise, for consistency and simplicity of specification.

A count to an ex command has been historically corrected to be no greater than the last line in a file; for example, in a five-line file, the command 1,6print would fail, but the command 1print300 would suc? ceed. POSIX.1?2008 requires conformance to historical practice.

Historically, the use of flags in ex commands could be obscure. General historical practice was as described by POSIX.1?2008, but there were some special cases. For instance, the list, number, and print commands ignored trailing address offsets; for example, 3p +++# would display line 3, and 3 would be the current line after the execution of the com? mand. The open and visual commands ignored both the trailing offsets

and the trailing flags. Also, flags specified to the open and visual commands interacted badly with the list edit option, and setting and then unsetting it during the open/visual session would cause vi to stop displaying lines in the specified format. For consistency and simplic? ity of specification, POSIX.1?2008 does not permit any of these excep? tions to the general rule.

POSIX.1?2008 uses the word copy in several places when discussing buf? fers. This is not intended to imply implementation.

Historically, ex users could not specify numeric buffers because of the ambiguity this would cause; for example, in the command 3 delete 2, it is unclear whether 2 is a buffer name or a count. POSIX.1?2008 re? quires conformance to historical practice by default, but does not pre? clude extensions.

Historically, the contents of the unnamed buffer were frequently dis? carded after commands that did not explicitly affect it; for example, when using the edit command to switch files. For consistency and sim? plicity of specification, POSIX.1?2008 does not permit this behavior.

The ex utility did not historically have access to the numeric buffers, and, furthermore, deleting lines in ex did not modify their contents. For example, if, after doing a delete in vi, the user switched to ex, did another delete, and then switched back to vi, the contents of the numeric buffers would not have changed. POSIX.1?2008 requires confor? mance to historical practice. Numeric buffers are described in the ex utility in order to confine the description of buffers to a single lo? cation in POSIX.1?2008.

The metacharacters that trigger shell expansion in file arguments match historical practice, as does the method for doing shell expansion. Im? plementations wishing to provide users with the flexibility to alter the set of metacharacters are encouraged to provide a shellmeta string edit option.

Historically, ex commands executed from vi refreshed the screen when it did not strictly need to do so; for example, :!date > /dev/null does not require a screen refresh because the output of the UNIX date com?

mand requires only a single line of the screen. POSIX.1?2008 requires that the screen be refreshed if it has been overwritten, but makes no requirements as to how an implementation should make that determina? tion. Implementations may prompt and refresh the screen regardless.

Abbreviate

Historical practice was that characters that were entered as part of an abbreviation replacement were subject to map expansions, the showmatch edit option, further abbreviation expansions, and so on; that is, they were logically pushed onto the terminal input queue, and were not a simple replacement. POSIX.1?2008 requires conformance to historical practice. Historical practice was that whenever a non-word character (that had not been escaped by a <control>?V) was entered after a word character, vi would check for abbreviations. The check was based on the type of the character entered before the word character of the word/non-word pair that triggered the check. The word character of the word/non-word pair that triggered the check and all characters entered before the trigger pair that were of that type were included in the check, with the exception of <blank> characters, which always delimited the abbreviation.

This means that, for the abbreviation to work, the lhs must end with a word character, there can be no transitions from word to non-word char? acters (or vice versa) other than between the last and next-to-last characters in the lhs, and there can be no <blank> characters in the lhs. In addition, because of the historical quoting rules, it was im? possible to enter a literal <control>?V in the lhs. POSIX.1?2008 re? quires conformance to historical practice. Historical implementations did not inform users when abbreviations that could never be used were entered; implementations are strongly encouraged to do so.

For example, the following abbreviations will work:

```
:ab (p  REPLACE
:ab p   REPLACE
:ab ((p REPLACE
```

The following abbreviations will not work:

:ab (REPLACE

:ab (pp REPLACE

Historical practice is that words on the vi colon command line were subject to abbreviation expansion, including the arguments to the ab?brev (and more interestingly) the unabbrev command. Because there are implementations that do not do abbreviation expansion for the first ar?gument to those commands, this is permitted, but not required, by POSIX.1?2008. However, the following sequence:

:ab foo bar

:ab foo baz

resulted in the addition of an abbreviation of "baz" for the string "bar" in historical ex/vi, and the sequence:

:ab foo1 bar

:ab foo2 bar

:unabbreviate foo2

deleted the abbreviation "foo1", not "foo2". These behaviors are not permitted by POSIX.1?2008 because they clearly violate the expectations of the user.

It was historical practice that <control>?V, not <backslash>, charac?ters be interpreted as escaping subsequent characters in the abbreviate command. POSIX.1?2008 requires conformance to historical practice; how?ever, it should be noted that an abbreviation containing a <blank> will never work.

Append

Historically, any text following a <vertical-line> command separator after an append, change, or insert command became part of the insert text. For example, in the command:

:g/pattern/append|stuff1

a line containing the text "stuff1" would be appended to each line matching pattern. It was also historically valid to enter:

:append|stuff1

stuff2

.

and the text on the ex command line would be appended along with the text inserted after it. There was an historical bug, however, that the user had to enter two terminating lines (the '.' lines) to terminate text input mode in this case. POSIX.1?2008 requires conformance to his? torical practice, but disallows the historical need for multiple termi? nating lines.

Change

See the RATIONALE for the append command. Historical practice for cur? sor positioning after the change command when no text is input, is as described in POSIX.1?2008. However, one System V implementation is known to have been modified such that the cursor is positioned on the first address specified, and not on the line before the first address. POSIX.1?2008 disallows this modification for consistency.

Historically, the change command did not support buffer arguments, al? though some implementations allow the specification of an optional buf? fer. This behavior is neither required nor disallowed by POSIX.1?2008.

Change Directory

A common extension in ex implementations is to use the elements of a cdpath edit option as prefix directories for path arguments to chdir that are relative pathnames and that do not have '.' or ".." as their first component. Elements in the cdpath edit option are <colon>-sepa? rated. The initial value of the cdpath edit option is the value of the shell CDPATH environment variable. This feature was not included in POSIX.1?2008 because it does not exist in any of the implementations considered historical practice.

Copy

Historical implementations of ex permitted copies to lines inside of the specified range; for example, :2,5copy3 was a valid command. POSIX.1?2008 requires conformance to historical practice.

Delete

POSIX.1?2008 requires support for the historical parsing of a delete command followed by flags, without any intervening <blank> characters. For example:

1dp Deletes the first line and prints the line that was second.

1delep As for 1dp.

1d 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.

Edit

Historically, any ex command could be entered as a +command argument to the edit command, although some (for example, insert and append) were known to confuse historical implementations. For consistency and sim?plicity of specification, POSIX.1?2008 requires that any command be supported as an argument to the edit command.

Historically, the command argument was executed with the current line set to the last line of the file, regardless of whether the edit com?mand was executed from visual mode or not. POSIX.1?2008 requires con?formance to historical practice.

Historically, the +command specified to the edit and next commands was delimited by the first <blank>, and there was no way to quote them. For consistency, POSIX.1?2008 requires that the usual ex backslash quoting be provided.

Historically, specifying the +command argument to the edit command re?quired a filename to be specified as well; for example, :edit +100 would always fail. For consistency and simplicity of specification, POSIX.1?2008 does not permit this usage to fail for that reason.

Historically, only the cursor position of the last file edited was re?membered by the editor. POSIX.1?2008 requires that this be supported; however, implementations are permitted to remember and restore the cur?sor position for any file previously edited.

File

Historical versions of the ex editor file command displayed a current line and number of lines in the edit buffer of 0 when the file was empty, while the vi <control>?G command displayed a current line and number of lines in the edit buffer of 1 in the same situation. POSIX.1?2008 does not permit this discrepancy, instead requiring that a

message be displayed indicating that the file is empty.

Global

The two-pass operation of the global and v commands is not intended to imply implementation, only the required result of the operation.

The current line and column are set as specified for the individual ex commands. This requirement is cumulative; that is, the current line and column must track across all the commands executed by the global or v commands.

Insert

See the RATIONALE for the append command.

Historically, insert could not be used with an address of zero; that is, not when the edit buffer was empty. POSIX.1?2008 requires that this command behave consistently with the append command.

Join

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

List

The historical output of the list command was potentially ambiguous. The standard developers believed correcting this to be more important than adhering to historical practice, and POSIX.1?2008 requires unam? biguous output.

Map

Historically, command mode maps only applied to command names; for ex? ample, if the character 'x' was mapped to 'y', the command fx searched for the 'x' character, not the 'y' character. POSIX.1?2008 requires this behavior. Historically, entering <control>?V as the first charac? ter of a vi command was an error. Several implementations have extended the semantics of vi such that <control>?V means that the subsequent command character is not mapped. This is permitted, but not required, by POSIX.1?2008. Regardless, using <control>?V to escape the second or later character in a sequence of characters that might match a map com?

mand, or any character in text input mode, is historical practice, and stops the entered keys from matching a map. POSIX.1?2008 requires con? formance to historical practice.

Historical implementations permitted digits to be used as a map command lhs, but then ignored the map. POSIX.1?2008 requires that the mapped digits not be ignored.

The historical implementation of the map command did not permit map commands that were more than a single character in length if the first character was printable. This behavior is permitted, but not required, by POSIX.1?2008.

Historically, mapped characters were remapped unless the remap edit op? tion was not set, or the prefix of the mapped characters matched the mapping characters; for example, in the map:

```
:map ab abcd
```

the characters "ab" were used as is and were not remapped, but the characters "cd" were mapped if appropriate. This can cause infinite loops in the vi mapping mechanisms. POSIX.1?2008 requires conformance to historical practice, and that such loops be interruptible.

Text input maps had the same problems with expanding the lhs for the ex map! and unmap! command as did the ex abbreviate and unabbreviate commands. See the RATIONALE for the ex abbreviate command. POSIX.1?2008 requires similar modification of some historical practice for the map and unmap commands, as described for the abbreviate and unabbreviate commands.

Historically, maps that were subsets of other maps behaved differently depending on the order in which they were defined. For example:

```
:map! ab     short
:map! abc    long
```

would always translate the characters "ab" to "short", regardless of how fast the characters "abc" were entered. If the entry order was re? versed:

```
:map! abc    long
:map! ab     short
```

the characters "ab" would cause the editor to pause, waiting for the completing 'c' character, and the characters might never be mapped to "short". For consistency and simplicity of specification, POSIX.1?2008 requires that the shortest match be used at all times.

The length of time the editor spends waiting for the characters to com? plete the lhs is unspecified because the timing capabilities of systems are often inexact and variable, and it may depend on other factors such as the speed of the connection. The time should be long enough for the user to be able to complete the sequence, but not long enough for the user to have to wait. Some implementations of vi have added a keytime option, which permits users to set the number of 0,1 seconds the editor waits for the completing characters. Because mapped terminal function and cursor keys tend to start with an <ESC> character, and <ESC> is the key ending vi text input mode, maps starting with <ESC> characters are generally exempted from this timeout period, or, at least timed out differently.

Mark

Historically, users were able to set the ``previous context'' marks ex? plicitly. In addition, the ex commands '' and '` and the vi commands '', ``, `', and '` all referred to the same mark. In addition, the pre? vious context marks were not set if the command, with which the address setting the mark was associated, failed. POSIX.1?2008 requires confor? mance to historical practice. Historically, if marked lines were deleted, the mark was also deleted, but would reappear if the change was undone. POSIX.1?2008 requires conformance to historical practice.

The description of the special events that set the ` and ' marks matches historical practice. For example, historically the command /a/,/b/ did not set the ` and ' marks, but the command /a/,/b/delete did.

Next

Historically, any ex command could be entered as a +command argument to the next command, although some (for example, insert and append) were known to confuse historical implementations. POSIX.1?2008 requires that

any command be permitted and that it behave as specified. The next com?
mand can accept more than one file, so usage such as:

next `ls [abc] `

is valid; it need not be valid for the edit or read commands, for exam?
ple, because they expect only one filename.

Historically, the next command behaved differently from the :rewind command in that it ignored the force flag if the autowrite flag was set. For consistency, POSIX.1?2008 does not permit this behavior.

Historically, the next command positioned the cursor as if the file had never been edited before, regardless. POSIX.1?2008 does not permit this behavior, for consistency with the edit command.

Implementations wanting to provide a counterpart to the next command that edited the previous file have used the command prev[ious], which takes no file argument. POSIX.1?2008 does not require this command.

Open

Historically, the open command would fail if the open edit option was not set. POSIX.1?2008 does not mention the open edit option and does not require this behavior. Some historical implementations do not per?
mit entering open mode from open or visual mode, only from ex mode. For consistency, POSIX.1?2008 does not permit this behavior.

Historically, entering open mode from the command line (that is, vi +open) resulted in anomalous behaviors; for example, the ex file and set commands, and the vi command <control>?G did not work. For consis?
tency, POSIX.1?2008 does not permit this behavior.

Historically, the open command only permitted '/' characters to be used as the search pattern delimiter. For consistency, POSIX.1?2008 requires that the search delimiters used by the s, global, and v commands be ac?
cepted as well.

Preserve

The preserve command does not historically cause the file to be consid?
ered unmodified for the purposes of future commands that may exit the editor. POSIX.1?2008 requires conformance to historical practice.

Historical documentation stated that mail was not sent to the user when

preserve was executed; however, historical implementations did send mail in this case. POSIX.1?2008 requires conformance to the historical implementations.

Print

The writing of NUL by the print command is not specified as a special case because the standard developers did not want to require ex to sup? port NUL characters. Historically, characters were displayed using the ARPA standard mappings, which are as follows:

1. Printable characters are left alone.
2. Control characters less than \177 are represented as '^' followed by the character offset from the '@' character in the ASCII map; for example, \007 is represented as '^G'.
3. \177 is represented as '^' followed by '?'.

The display of characters having their eighth bit set was less stan? dard. Existing implementations use hex (0x00), octal (\000), and a meta-bit display. (The latter displayed bytes that had their eighth bit set as the two characters "M-" followed by the seven-bit display as de? scribed above.) The latter probably has the best claim to historical practice because it was used for the -v option of 4 BSD and 4 BSD-de? rived versions of the cat utility since 1980.

No specific display format is required by POSIX.1?2008.

Explicit dependence on the ASCII character set has been avoided where possible, hence the use of the phrase an ``implementation-defined multi-character sequence'' for the display of non-printable characters in preference to the historical usage of, for instance, "^I" for the <tab>. Implementations are encouraged to conform to historical prac? tice in the absence of any strong reason to diverge.

Historically, all ex commands beginning with the letter 'p' could be entered using capitalized versions of the commands; for example, P[rint], Pre[serve], and Pu[t] were all valid command names. POSIX.1?2008 permits, but does not require, this historical practice because capital forms of the commands are used by some implementations for other purposes.

Put

Historically, an ex put command, executed from open or visual mode, was the same as the open or visual mode P command, if the buffer was named and was cut in character mode, and the same as the p command if the buffer was named and cut in line mode. If the unnamed buffer was the source of the text, the entire line from which the text was taken was usually put, and the buffer was handled as if in line mode, but it was possible to get extremely anomalous behavior. In addition, using the Q command to switch into ex mode, and then doing a put often resulted in errors as well, such as appending text that was unrelated to the (sup? posed) contents of the buffer. For consistency and simplicity of speci? fication, POSIX.1?2008 does not permit these behaviors. All ex put com? mands are required to operate in line mode, and the contents of the buffers are not altered by changing the mode of the editor.

Read

Historically, an ex read command executed from open or visual mode, ex? ecuted in an empty file, left an empty line as the first line of the file. For consistency and simplicity of specification, POSIX.1?2008 does not permit this behavior. Historically, a read in open or visual mode from a program left the cursor at the last line read in, not the first. For consistency, POSIX.1?2008 does not permit this behavior.

Historical implementations of ex were unable to undo read commands that read from the output of a program. For consistency, POSIX.1?2008 does not permit this behavior.

Historically, the ex and vi message after a successful read or write command specified ``characters'', not ``bytes''. POSIX.1?2008 requires that the number of bytes be displayed, not the number of characters, because it may be difficult in multi-byte implementations to determine the number of characters read. Implementations are encouraged to clar? ify the message displayed to the user.

Historically, reads were not permitted on files other than type regu? lar, except that FIFO files could be read (probably only because they did not exist when ex and vi were originally written). Because the his?

torical ex evaluated read! and read ! equivalently, there can be no optional way to force the read. POSIX.1?2008 permits, but does not re? quire, this behavior.

Recover

Some historical implementations of the editor permitted users to re? cover the edit buffer contents from a previous edit session, and then exit without saving those contents (or explicitly discarding them). The intent of POSIX.1?2008 in requiring that the edit buffer be treated as already modified is to prevent this user error.

Rewind

Historical implementations supported the rewind command when the user was editing the first file in the list; that is, the file that the rewind command would edit. POSIX.1?2008 requires conformance to histor? ical practice.

Substitute

Historically, ex accepted an r option to the s command. The effect of the r option was to use the last regular expression used in any command as the pattern, the same as the ~ command. The r option is not required by POSIX.1?2008. Historically, the c and g options were toggled; for example, the command :s/abc/def/ was the same as s/abc/def/ccccgggg. For simplicity of specification, POSIX.1?2008 does not permit this be? havior.

The tilde command is often used to replace the last search RE. For ex? ample, in the sequence:

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

the ~ command is equivalent to:

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

Historically, ex accepted all of the following forms:

```
s/abc/def/
s/abc/def
s/abc/
```

s/abc

POSIX.1?2008 requires conformance to this historical practice.

The s command presumes that the '^' character only occupies a single column in the display. Much of the ex and vi specification presumes that the <space> only occupies a single column in the display. There are no known character sets for which this is not true.

Historically, the final column position for the substitute commands was based on previous column movements; a search for a pattern followed by a substitution would leave the column position unchanged, while a 0 command followed by a substitution would change the column position to the first non-<blank>. For consistency and simplicity of specifica? tion, POSIX.1?2008 requires that the final column position always be set to the first non-<blank>.

Set

Historical implementations redisplayed all of the options for each oc? currence of the all keyword. POSIX.1?2008 permits, but does not re? quire, this behavior.

Tag

No requirement is made as to where ex and vi shall look for the file referenced by the tag entry. Historical practice has been to look for the path found in the tags file, based on the current directory. A use? ful extension found in some implementations is to look based on the di? rectory containing the tags file that held the entry, as well. No re? quirement is made as to which reference for the tag in the tags file is used. This is deliberate, in order to permit extensions such as multi? ple entries in a tags file for a tag.

Because users often specify many different tags files, some of which need not be relevant or exist at any particular time, POSIX.1?2008 re? quires that error messages about problem tags files be displayed only if the requested tag is not found, and then, only once for each time that the tag edit option is changed.

The requirement that the current edit buffer be unmodified is only nec? essary if the file indicated by the tag entry is not the same as the

current file (as defined by the current pathname). Historically, the file would be reloaded if the filename had changed, as well as if the filename was different from the current pathname. For consistency and simplicity of specification, POSIX.1?2008 does not permit this behav? ior, requiring that the name be the only factor in the decision.

Historically, vi only searched for tags in the current file from the current cursor to the end of the file, and therefore, if the wrapscan option was not set, tags occurring before the current cursor were not found. POSIX.1?2008 considers this a bug, and implementations are re? quired to search for the first occurrence in the file, regardless.

Undo

The undo description deliberately uses the word ``modified''. The undo command is not intended to undo commands that replace the contents of the edit buffer, such as edit, next, tag, or recover.

Cursor positioning after the undo command was inconsistent in the his? torical vi, sometimes attempting to restore the original cursor posi? tion (global, undo, and v commands), and sometimes, in the presence of maps, placing the cursor on the last line added or changed instead of the first. POSIX.1?2008 requires a simplified behavior for consistency and simplicity of specification.

Version

The version command cannot be exactly specified since there is no widely-accepted definition of what the version information should con? tain. Implementations are encouraged to do something reasonably intel? ligent.

Write

Historically, the ex and vi message after a successful read or write command specified ``characters'', not ``bytes''. POSIX.1?2008 requires that the number of bytes be displayed, not the number of characters be? cause it may be difficult in multi-byte implementations to determine the number of characters written. Implementations are encouraged to clarify the message displayed to the user.

Implementation-defined tests are permitted so that implementations can

make additional checks; for example, for locks or file modification times.

Historically, attempting to append to a nonexistent file caused an er? ror. It has been left unspecified in POSIX.1?2008 to permit implementa? tions to let the write succeed, so that the append semantics are simi? lar to those of the historical csh.

Historical vi permitted empty edit buffers to be written. However, since the way vi got around dealing with ``empty'' files was to always have a line in the edit buffer, no matter what, it wrote them as files of a single, empty line. POSIX.1?2008 does not permit this behavior.

Historically, ex restored standard output and standard error to their values as of when ex was invoked, before writes to programs were per? formed. This could disturb the terminal configuration as well as be a security issue for some terminals. POSIX.1?2008 does not permit this, requiring that the program output be captured and displayed as if by the ex print command.

Adjust Window

Historically, the line count was set to the value of the scroll option if the type character was end-of-file. This feature was broken on most historical implementations long ago, however, and is not documented anywhere. For this reason, POSIX.1?2008 is resolutely silent.

Historically, the z command was <blank>-sensitive and z + and z - did different things than z+ and z- because the type could not be distin? guished from a flag. (The commands z . and z = were historically in? valid.) POSIX.1?2008 requires conformance to this historical practice.

Historically, the z command was further <blank>-sensitive in that the count could not be <blank>-delimited; for example, the commands z= 5 and z- 5 were also invalid. Because the count is not ambiguous with re? spect to either the type character or the flags, this is not permitted by POSIX.1?2008.

Escape

Historically, ex filter commands only read the standard output of the commands, letting standard error appear on the terminal as usual. The

vi utility, however, read both standard output and standard error. POSIX.1?2008 requires the latter behavior for both ex and vi, for con? sistency.

Shift Left and Shift Right

Historically, it was possible to add shift characters to increase the effect of the command; for example, <<< outdented (or >>> indented) the lines 3 levels of indentation instead of the default 1. POSIX.1?2008 requires conformance to historical practice.

<control>?D

Historically, the <control>?D command erased the prompt, providing the user with an unbroken presentation of lines from the edit buffer. This is not required by POSIX.1?2008; implementations are encouraged to pro? vide it if possible. Historically, the <control>?D command took, and then ignored, a count. POSIX.1?2008 does not permit this behavior.

Write Line Number

Historically, the ex = command, when executed in ex mode in an empty edit buffer, reported 0, and from open or visual mode, reported 1. For consistency and simplicity of specification, POSIX.1?2008 does not per? mit this behavior.

Execute

Historically, ex did not correctly handle the inclusion of text input commands (that is, append, insert, and change) in executed buffers. POSIX.1?2008 does not permit this exclusion for consistency.

Historically, the logical contents of the buffer being executed did not change if the buffer itself were modified by the commands being exe? cuted; that is, buffer execution did not support self-modifying code. POSIX.1?2008 requires conformance to historical practice.

Historically, the @ command took a range of lines, and the @ buffer was executed once per line, with the current line ('.') set to each speci? fied line. POSIX.1?2008 requires conformance to historical practice.

Some historical implementations did not notice if errors occurred dur? ing buffer execution. This, coupled with the ability to specify a range of lines for the ex @ command, makes it trivial to cause them to drop

core. POSIX.1?2008 requires that implementations stop buffer execution if any error occurs, if the specified line doesn't exist, or if the contents of the edit buffer itself are replaced (for example, the buf? fer executes the ex :edit command).

Regular Expressions in ex

Historical practice is that the characters in the replacement part of the last s command?that is, those matched by entering a '~' in the reg? ular expression?were not further expanded by the regular expression en? gine. So, if the characters contained the string "a.," they would match 'a' followed by "." and not 'a' followed by any character. POSIX.1?2008 requires conformance to historical practice.

Edit Options in ex

The following paragraphs describe the historical behavior of some edit options that were not, for whatever reason, included in POSIX.1?2008. Implementations are strongly encouraged to only use these names if the functionality described here is fully supported.

extended The extended edit option has been used in some implementa? tions of vi to provide extended regular expressions instead of basic regular expressions This option was omitted from POSIX.1?2008 because it is not widespread historical prac? tice.

flash The flash edit option historically caused the screen to flash instead of beeping on error. This option was omitted from POSIX.1?2008 because it is not found in some historical im? plementations.

hardtabs The hardtabs edit option historically defined the number of columns between hardware tab settings. This option was omit? ted from POSIX.1?2008 because it was believed to no longer be generally useful.

modeline The modeline (sometimes named modelines) edit option histori? cally caused ex or vi to read the five first and last lines of the file for editor commands. This option is a security problem, and vendors are strongly encouraged to delete it

from historical implementations.

open The open edit option historically disallowed the ex open and visual commands. This edit option was omitted because these commands are required by POSIX.1?2008.

optimize The optimize edit option historically expedited text through? put by setting the terminal to not do automatic <carriage-re? turn> characters when printing more than one logical line of output. This option was omitted from POSIX.1?2008 because it was intended for terminals without addressable cursors, which are rarely, if ever, still used.

ruler The ruler edit option has been used in some implementations of vi to present a current row/column ruler for the user. This option was omitted from POSIX.1?2008 because it is not widespread historical practice.

sourceany The sourceany edit option historically caused ex or vi to source start-up files that were owned by users other than the user running the editor. This option is a security problem, and vendors are strongly encouraged to remove it from their implementations.

timeout The timeout edit option historically enabled the (now stan? dard) feature of only waiting for a short period before re? turning keys that could be part of a macro. This feature was omitted from POSIX.1?2008 because its behavior is now stan? dard, it is not widely useful, and it was rarely documented.

verbose The verbose edit option has been used in some implementations of vi to cause vi to output error messages for common errors; for example, attempting to move the cursor past the beginning or end of the line instead of only alerting the screen. (The historical vi only alerted the terminal and presented no mes? sage for such errors. The historical editor option terse did not select when to present error messages, it only made ex? isting error messages more or less verbose.) This option was omitted from POSIX.1?2008 because it is not widespread his?

torical practice; however, implementors are encouraged to use it if they wish to provide error messages for naive users.

wraplen The wraplen edit option has been used in some implementations of vi to specify an automatic margin measured from the left margin instead of from the right margin. This is useful when multiple screen sizes are being used to edit a single file. This option was omitted from POSIX.1?2008 because it is not widespread historical practice; however, implementors are en? couraged to use it if they add this functionality.

autoindent, ai

Historically, the command 0a did not do any autoindentation, regardless of the current indentation of line 1. POSIX.1?2008 requires that any indentation present in line 1 be used.

autoprint, ap

Historically, the autoprint edit option was not completely consistent or based solely on modifications to the edit buffer. Exceptions were the read command (when reading from a file, but not from a filter), the append, change, insert, global, and v commands, all of which were not affected by autoprint, and the tag command, which was affected by auto? print. POSIX.1?2008 requires conformance to historical practice. Historically, the autoprint option only applied to the last of multiple commands entered using <vertical-line> delimiters; for example, delete <newline> was affected by autoprint, but delete|version <newline> was not. POSIX.1?2008 requires conformance to historical practice.

autowrite, aw

Appending the '!' character to the ex next command to avoid performing an automatic write was not supported in historical implementations. POSIX.1?2008 requires that the behavior match the other ex commands for consistency.

ignorecase, ic

Historical implementations of case-insensitive matching (the ignorecase edit option) lead to counter-intuitive situations when uppercase char? acters were used in range expressions. Historically, the process was as

follows:

1. Take a line of text from the edit buffer.

2. Convert uppercase to lowercase in text line.

3. Convert uppercase to lowercase in regular expressions, except in character class specifications.

4. Match regular expressions against text.

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

The cat sat on the mat

would be matched by

/^the/

but not by:

/^[A-Z]he/

For consistency with other commands implementing regular expressions, POSIX.1?2008 does not permit this behavior.

paragraphs, para

The ISO POSIX?2:1993 standard made the default paragraphs and sections edit options implementation-defined, arguing they were historically oriented to the UNIX system troff text formatter, and a ``portable user'' could use the {, }, [[,]], (, and) commands in open or visual mode and have the cursor stop in unexpected places. POSIX.1?2008 speci? fies their values in the POSIX locale because the unusual grouping (they only work when grouped into two characters at a time) means that they cannot be used for general-purpose movement, regardless.

readonly

Implementations are encouraged to provide the best possible information to the user as to the read-only status of the file, with the exception that they should not consider the current special privileges of the process. This provides users with a safety net because they must force the overwrite of read-only files, even when running with additional privileges.

The readonly edit option specification largely conforms to historical practice. The only difference is that historical implementations did not notice that the user had set the readonly edit option in cases

where the file was already marked read-only for some reason, and would therefore reinitialize the readonly edit option the next time the con? tents of the edit buffer were replaced. This behavior is disallowed by POSIX.1?2008.

report

The requirement that lines copied to a buffer interact differently than deleted lines is historical practice. For example, if the report edit option is set to 3, deleting 3 lines will cause a report to be written, but 4 lines must be copied before a report is written.

The requirement that the ex global, v, open, undo, and visual commands present reports based on the total number of lines added or deleted during the command execution, and that commands executed by the global and v commands not present reports, is historical practice. POSIX.1?2008 extends historical practice by requiring that buffer exe? cution be treated similarly. The reasons for this are two-fold. Histor? ically, only the report by the last command executed from the buffer would be seen by the user, as each new report would overwrite the last. In addition, the standard developers believed that buffer execution had more in common with global and v commands than it did with other ex commands, and should behave similarly, for consistency and simplicity of specification.

showmatch, sm

The length of time the cursor spends on the matching character is un? specified because the timing capabilities of systems are often inexact and variable. The time should be long enough for the user to notice, but not long enough for the user to become annoyed. Some implementa? tions of vi have added a matchtime option that permits users to set the number of 0,1 second intervals the cursor pauses on the matching char? acter.

showmode

The showmode option has been used in some historical implementations of ex and vi to display the current editing mode when in open or visual mode. The editing modes have generally included ``command'' and ``in?

put", and sometimes other modes such as ``replace'' and ``change''.

The string was usually displayed on the bottom line of the screen at the far right-hand corner. In addition, a preceding '*' character often denoted whether the contents of the edit buffer had been modified. The latter display has sometimes been part of the showmode option, and sometimes based on another option. This option was not available in the 4 BSD historical implementation of vi, but was viewed as generally use? ful, particularly to novice users, and is required by POSIX.1?2008.

The smd shorthand for the showmode option was not present in all his? torical implementations of the editor. POSIX.1?2008 requires it, for consistency.

Not all historical implementations of the editor displayed a mode string for command mode, differentiating command mode from text input mode by the absence of a mode string. POSIX.1?2008 permits this behav? ior for consistency with historical practice, but implementations are encouraged to provide a display string for both modes.

slowopen

Historically, the slowopen option was automatically set if the terminal baud rate was less than 1200 baud, or if the baud rate was 1200 baud and the redraw option was not set. The slowopen option had two effects. First, when inserting characters in the middle of a line, characters after the cursor would not be pushed ahead, but would appear to be overwritten. Second, when creating a new line of text, lines after the current line would not be scrolled down, but would appear to be over? written. In both cases, ending text input mode would cause the screen to be refreshed to match the actual contents of the edit buffer. Fi? nally, terminals that were sufficiently intelligent caused the editor to ignore the slowopen option. POSIX.1?2008 permits most historical be? havior, extending historical practice to require slowopen behaviors if the edit option is set by the user.

tags

The default path for tags files is left unspecified as implementations may have their own tags implementations that do not correspond to the

historical ones. The default tags option value should probably at least include the file ./tags.

term

Historical implementations of ex and vi ignored changes to the term edit option after the initial terminal information was loaded. This is permitted by POSIX.1?2008; however, implementations are encouraged to permit the user to modify their terminal type at any time.

terse

Historically, the terse edit option optionally provided a shorter, less descriptive error message, for some error messages. This is permitted, but not required, by POSIX.1?2008. Historically, most common visual mode errors (for example, trying to move the cursor past the end of a line) did not result in an error message, but simply alerted the termi? nal. Implementations wishing to provide messages for novice users are urged to do so based on the edit option verbose, and not terse.

window

In historical implementations, the default for the window edit option was based on the baud rate as follows:

1. If the baud rate was less than 1200, the edit option w300 set the window value; for example, the line:

   set w300=12

   would set the window option to 12 if the baud rate was less than 1200.

2. If the baud rate was equal to 1200, the edit option w1200 set the window value.

3. If the baud rate was greater than 1200, the edit option w9600 set the window value.

The w300, w1200, and w9600 options do not appear in POSIX.1?2008 be? cause of their dependence on specific baud rates.

In historical implementations, the size of the window displayed by var? ious commands was related to, but not necessarily the same as, the win? dow edit option. For example, the size of the window was set by the ex command visual 10, but it did not change the value of the window edit

option. However, changing the value of the window edit option did change the number of lines that were displayed when the screen was re? painted. POSIX.1?2008 does not permit this behavior in the interests of consistency and simplicity of specification, and requires that all com? mands that change the number of lines that are displayed do it by set? ting the value of the window edit option.

wrapmargin, wm

Historically, the wrapmargin option did not affect maps inserting char? acters that also had associated counts; for example :map K 5aABC DEF. Unfortunately, there are widely used maps that depend on this behavior. For consistency and simplicity of specification, POSIX.1?2008 does not permit this behavior.

Historically, wrapmargin was calculated using the column display width of all characters on the screen. For example, an implementation using "^I" to represent <tab> characters when the list edit option was set, where '^' and 'I' each took up a single column on the screen, would calculate the wrapmargin based on a value of 2 for each <tab>. The number edit option similarly changed the effective length of the line as well. POSIX.1?2008 requires conformance to historical practice.

Earlier versions of this standard allowed for implementations with bytes other than eight bits, but this has been modified in this ver? sion.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.9.1.1, Command Search and Execution, ctags, ed, sed, sh, stty, vi

The Base Definitions volume of POSIX.1?2017, Table 5-1, Escape Se? quences and Associated Actions, Chapter 8, Environment Variables, Sec? tion 9.3, Basic Regular Expressions, Section 12.2, Utility Syntax Guidelines

The System Interfaces volume of POSIX.1?2017, access()