Ed Text Editor Summary

Line Addressing Summary:

Address Symbol	Description
.	The current line (address) in the buffer.
$	The last line in the buffer.
n	The nth, line in the buffer where n is a number in the range [**0**,**$**].
- **^**	The previous line. This is equivalent to **-1** and may be repeated with cumulative effect.
-n **^n**	The nth previous line, where n is a non-negative number.
+	The next line. This is equivalent to **+1** and may be repeated with cumulative effect.
+n **whitespace n**	The nth next line, where n is a non-negative number. Whitespace followed by a number n is interpreted as **+n**.
, **%**	The first through last lines in the buffer. This is equivalent to the address range **1**,**$**.
;	The current through last lines in the buffer. This is equivalent to the address range **.**,**$**.
/re/	The next line containing the regular expression ***re***. The search wraps to the beginning of the buffer and continues down to the current line, if necessary. // repeats the last search.
?re?	The previous line containing the regular expression ***re***. The search wraps to the end of the buffer and continues up to the current line, if necessary. **??** repeats the last search.
´lc	The line previously marked by a \`**k**' (mark) command, where ***lc*** is a lower case letter.

Each address in a comma-delimited range is interpreted relative to the current address.
In a semicolon-delimited range, the 1st address is used to set the current address, and the 2nd address is interpreted relative to the first.

Command Summary:

Command	Description
(.)a	Appends text to the buffer after the addressed line, which may be the address 0 (zero). Text is entered in input mode. The current address is set to last line entered.
(.,.)c	Changes lines in the buffer. The addressed lines are deleted from the buffer, and text is appended in their place. Text is entered in input mode. The current address is set to last line entered.
(.,.)d	Deletes the addressed lines from the buffer. If there is a line after the deleted range, then the current address is set to this line. Otherwise the current address is set to the line before the deleted range.
e *file*	Edits ***file***, and sets the default filename. If ***file*** is not specified, then the default filename is used. Any lines in the buffer are deleted before the new file is read. The current address is set to the last line read.
e !*command*	Edits the standard output of \`!***command***', (see **!command** below). The default filename is unchanged. Any lines in the buffer are deleted before the output of command is read. The current address is set to the last line read.
E *file*	Edits ***file*** unconditionally. This is similar to the **e** command, except that unwritten changes are discarded without warning. The current address is set to the last line read.
f *file*	Sets the default filename to file. If file is not specified, then the default unescaped filename is printed.
(1,$)g*/re/command-list*	Applies command-list to each of the addressed lines matching a regular expression ***re***. The current address is set to the line currently matched before ***command-list*** is executed. At the end of the \`**g**' command, the current address is set to the last line affected by command-list.  Each command in ***command-list*** must be on a separate line, and every line except for the last must be terminated by a back-slash (\\). Any commands are allowed, except for \`**g**', \`**G**', \`**v**', and \`**V**'. A newline alone in command-list is equivalent to a \`**p**' command.
(1,$)G*/re/*	Interactively edits the addressed lines matching a regular expression ***re***. For each matching line, the line is printed, the current address is set, and the user is prompted to enter a command-list. At the end of the \`**G**' command, the current address is set to the last line affected by (the last) command-list.  The format of command-list is the same as that of the \`**g**' command. A newline alone acts as a null command list. A single \`**&**' repeats the last non-null command list.
H	Toggles the printing of error explanations. By default, explanations are not printed. It is recommended that *ed* scripts begin with this command to aid in debugging.
h	Prints an explanation of the last error.
(.)i	Inserts text in the buffer before the current line. Text is entered in input mode. The current address is set to the last line entered.

Command (continued)	Description
(.,.+1)j	Joins the addressed lines. The addressed lines are deleted from the buffer and replaced by a single line containing their joined text. The current address is set to the resultant line.
(.)k*lc*	Marks a line with a lower case letter ***lc***. The line can then be addressed as **'*lc*** (i.e., a single quote followed by ***lc***) in subsequent commands. The mark is not cleared until the line is deleted or otherwise modified.
(.,.)l	Prints the addressed lines unambiguously. If invoked from a terminal, *ed* pauses at the end of each page until a newline is entered. The current address is set to the last line printed.
(.,.)m(.)	Moves lines in the buffer. The addressed lines are moved to after the right-hand destination address, which may be the address 0 (zero). The current address is set to the last line moved.
(.,.)n	Prints the addressed lines along with their line nums. The curr. addr is set to the last line printed
(.,.)p	Prints the addressed lines. If invoked from a terminal, *ed* pauses at the end of each page until a newline is entered. The current address is set to the last line printed.
P	Toggles the command prompt on and off. Unless a prompt was specified by with command-line option -p string, the command prompt is by default turned off.
q	Quits ed.
Q	Quits ed unconditionally. This is similar to the **q** command, except that unwritten changes are discarded without warning.
($)r *file*	Reads ***file*** to after the addressed line. If ***file*** is not specified, then the default filename is used. If there was no default filename prior to the command, then the default filename is set to ***file***. Otherwise, the default filename is unchanged. The current address is set to the last line read.
($)r !*command*	Reads to after the addressed line the standard output of **`!*command*'**, (see the **!command** below). The default filename is unchanged. The current address is set to the last line read.
(.,.)s/*re*/*replacement*/ (.,.)s/*re*/*replacement*/g (.,.)s/*re*/*replacement*/n	Replaces text in the addressed lines matching a regular expression ***re*** with ***replacement***. By default, only the first match in each line is replaced. If the **`g'** (global) suffix is given, then every match to be replaced. The **`n'** suffix, where n is a positive number, causes only the n-th match to be replaced. It is an error if no substitutions are performed on any of the addressed lines. The current address is set the last line affected. ***re*** and ***replacement*** may be delimited by any character other than space and newline (see the **`s'** command below). If one or two of the last delimiters is omitted, then the last line affected is printed as though the print suffix **`p'** were specified. An unescaped **`&'** in replacement is replaced by the currently matched text. The character sequence **`\m'**, where **m** is a number in the range [1,9], is replaced by the **m**-th backreference expression of the matched text. If replacement consists of a single **`%'**, then replacement from the last substitution is used. Newlines may be embedded in replacement if they are escaped with a backslash (\).
(.,.)s	Repeats the last substitution. This form of the **`s'** command accepts a count suffix **`n'**, or any combination of the characters **`r'**, **`g'**, and **`p'**. If a count suffix **`n'** is given, then only the **n**-th match is replaced. The **`r'** suffix causes the regular expression of the last search to be used instead of that of the last substitution. The **`g'** suffix toggles the global suffix of the last substitution. The **`p'** suffix toggles the print suffix of the last substitution. The current address is set to the last line affected.
(.,.)t(.)	Copies (i.e., **t**ransfers) the addressed lines to after the right-hand destination address, which may be the address 0 (zero). The current address is set to the last line copied.
u	Undoes the last command and restores the current address to what it was before the command. The global commands **`g'**, **`G'**, **`v'**, and **`V'** are treated as a single command by undo. **`u'** is its own inverse.
(1,$)v/*re*/*command-list*	Applies ***command-list*** to each of the addressed lines not matching a regular expression ***re***. This is similar to the **`g'** command.
(1,$)V/*re*/	Interactively edits the addressed lines not matching a regular expression ***re***. This is similar to the **`G'** command.
(1,$)w *file*	Writes the addressed lines to ***file***. Any previous contents of ***file*** is lost without warning. If there is no default filename, then the default filename is set to ***file***, otherwise it is unchanged. If no filename is specified, then the default filename is used. The current address is unchanged.
(1,$)wq *file*	Writes the addressed lines to ***file***, and then executes a **`q'** command.
(1,$)w !*command*	Writes the addressed lines to the standard input of **`!*command*'**, (see the **!command** below). The default filename and current address are unchanged.
(1,$)W *file*	Appends the addressed lines to the end of ***file***. This is similar to the **`w'** command, expect that the previous contents of file is not clobbered. The current address is unchanged.
(.)x	Copies (puts) the contents of the cut buffer to after the addressed line. The current address is set to the last line copied.
(.,.)y	Copies (yanks) the addressed lines to the cut buffer. The cut buffer is overwritten by subsequent **`y'**, **`s'**, **`j'**, **`d'**, or **`c'** commands. The current address is unchanged.
(.+1)z*n*	Scrolls ***n*** lines at a time starting at addressed line. If ***n*** is not specified, then the current window size is used. The current address is set to the last line printed.
!*command*	Executes command via *sh*(1). If the first character of command is **`!'**, then it is replaced by text of the previous **`!*command*'**. *ed* does not process command for backslash (\) escapes. However, an unescaped **`%'** is replaced by the default filename. When the shell returns from execution, a **`!'** is printed to the standard output. The current line is unchanged.
(.,.)#	Begins a comment; the rest of the line, up to a newline, is ignored. If a line address followed by a semicolon is given, then the current address is set to that address. Otherwise, the current address is unchanged.
($)=	Prints the line number of the addressed line.
(.+1)newline	Prints the addressed line, and sets the current address to that line.