NAME

e – the RAND editor, E

SYNOPSIS

e [options] [mainfile [altfile]]

DESCRIPTION

The Rand editor *E* is a full screen editor which treats a text file as a two-dimensional surface with text on it. Among its more unusual features are manipulating rectangular areas of text, editing several files at once, and displaying text in one or more windows. While it is not as powerful as *Emacs,* or as widely used as *Vi,* most people at Rand preferred its easy to learn commands and used it to edit both programs and text, during the period ranging from roughly 1973 to 1995. *E* is a complete rewrite of the Rand *NED* editor of many years back. The current version (E19+) has been improved to take advantage of terminals that have the ability to do local insert/delete line, and it can now use *termcap* or *terminfo* so that most terminals with direct cursor addressing can be used.

Please note that this manual entry does not try to be either a tutorial or a full reference manual for the editor. Rand users should initially learn *E* by reading the RAND document *Self Teaching Guide to RAND's Text Processor, 1986,* available from *https://rand.org* as N-2056-1. Report N-2239-1, *The RAND Editor e: Version 19, 1986,* is a full reference manual for the editor, and is mostly correct for this version. Two other useful reports are R-2000-ARPA, *A Guide to NED: A New On-Line Computer Editor, 1977* and R-2176-ARPA, *The CRT Text Editor NED -- Introduction and Reference Manual, 1977.*

The *mainfile* on the command line is the file which is initially displayed in the editor. If the file doesn't exist, the user will be asked if it should be created. If in addition, an *altfile* is specified on the command line, it will be read in as the initial alternate file. If no filenames are present on the command line, *e* will attempt to continue from your previous editing session. To do this it saves "state" in a file called '.es1' in your current working directory. Before invoking the editor, be sure and specify your terminal type by setting the TERM environment variable.

Refer to section RECENT CHANGES for a list of new changes to E.

OPTIONS

The currently available command line options are:

−help

This causes a display of the currently available options and their values along with the version number of the editor.

−bullets or **−nobullets**

Options −bullets and −nobullets forces border bullets on or off regardless of line speed or terminal type. Line speeds of 1200 baud and slower and some terminal types set border bullets off in the absence of these options. "Border bullets" does not refer to the characters making up the border of the editing window in "e". They refer to four characters placed at the top, bottom, left and right of the current window indicating the current postion of the cursor. They are chiefly useful on terminals or terminal emulators which do not have a blinking cursor, but some people may prefer them as an added indication of the cursor position. If the terminal supports highlighting, they will be highlighted.

−create

If the file named on the command line does not exist, the editor will ordinarily ask if you want to create it. This option causes the editor to create the file immediately, without asking.

−newfilemode=mode

If the editor creates a new file, it will be created with mode "mode". This is of the typical UNIX octal form, e.g. "0644".

−debug=filename

The −debug option causes the editor to write debugging information to the file "filename".

−inplace

Option −inplace preserves file links except for singly−linked files. From within the editor, link preservation can be turned on and off, even for singly−linked files, by the update command.

−terminal=termtype

Option −terminal=termtype identifies the terminal type explicitly. In the absence of the −terminal option the terminal type is retrieved from the TERM environment variable.

If there is no compiled−in code for the terminal, the termcap entry for the terminal type will be used and the "standard" keyboard will be assigned. Refer to TERMINALS SUPPORTED section of this document for terminals that can be supported by compiled−in code (varies with installation).

−keyboard=termtype
−dkeyboard

Option −keyboard=termtype sets the keyboard type to "termtype". It overrides any "−terminal=" argument and also any EKBD environment variable. Option −dkeyboard sets the keyboard type to "standard". This is a shortcut for "−keyboard=standard".

−dtermcap

Option −dtermcap forces *E* to use the termcap entry for the terminal identified by the TERM variable or −terminal option, disregarding any applicable compiled-in code.

−notracks

Option −notracks will allow you to edit a file without using or disturbing the work files (keys, state, and change files) from your previous session.

−replay=filename

Option −replay=filename allows you to replay with a specified keys file, where "filename" represents that keys file. Use −silent when the replay display is not desired. The message "***replay completed" or "***replay aborted" signals the end of the replay. The <INT> key can be used to stop the replay short of completion. Files .ec1* and .ek1* must not be present if the replay is to work.

−norecover

Option −norecover will open the edit session as defined by the .es1 state file, ignoring any recovery processing of a prior crashed or aborted session.

−regexp

Option −regexp enables regular expression pattern matching for searching and replacing. This mode is also enabled by the "re[gexp]" command. In regular expression mode, "RE" is displayed at the bottom of the screen. The command "−re[gexp]" returns to non-regular expression mode. The regular expressions recognized are essentially the same as that for *ed*.

−state=filename

Option −state=filename causes the initial state of the editor to be read from the indicated "state" file. The state file from a previous *E* session (with alternate files, windows, cursor positioning, etc.) may be saved by exiting, and renaming the state file (".es1").

−[no]stick

Option −stick sets the behavior at the right margin to stop in response to text or a <RIGHT ARROW>. −nostick is the default, and will automatically shift the window right in response to text or the <RIGHT ARROW> key.

−readonly

Option −readonly will prohibit any changes to the file(s).

−noreadonly

Option −noreadonly will disable the (new e19) mechanism preventing changes to files lacking write permission, that you could previously change because you had write permission in the directory.

−noprofile
Option −noprofile will prevent reading of the ".e_profile". (See *caveat* under STARTUP FILE.

−profile=file
Specifies the profile to be used in lieu of the default "./.e_profile" or "~/.e_profile". See STARTUP FILE in a later section.

−blanks
Expands tabs to blanks. The keyletters BL are be displayed at the bottom of the screen.

−literal
Displays tabs in ^I notation. The keyletters LIT are displayed at the bottom of the screen.

−tabs
Don't convert tabs to blanks. The keyletters TAB are displayed at the bottom of the screen.

−nostrip
Do not remove blanks at the end of lines. The keyletters NOS are displayed at the bottom of the screen.

−nolocks
Turns off file locking if the editor was compiled with the option FILELOCKING.

−nomacros
Prevents reading a ".e_macros" file if one exists. See MACROS in a later section.

−search=string
Sets the initial search string.

NEW OPTIONS IN E20

−nowinshift
In previous editions of the editor, creating a new window on the current file by typing "CMD: window" would obscure one line of text in the file with the top border of the new window. This avoided the necessity of repainting the text in the new window by shifting it down, which was desirable in the days of slow serial lines. The current version of the editor shifts the new window down by one line, to allow all of the text in the file to be visible in one window or the other. Option −nowinshift causes the editor to revert to the old behavior.

−noresize
The editor now supports resizing of the terminal window in which it's running. It will resize its own windows to match. There are a few situations in which this adaptive behavior is undesirable. This option prevents the editor from resizing its own windows when the terminal window in which it's running is resized.

These next four options control the foreground and background colors used to highlight the areas selected by the MARK command. If you have 256 colors available, some values to try for the **−setab=N** option are: 3, 7, 87, 136, 151, 194, and 222.

See MOUSE SUPPORT for details.

−bgcolor=r,g,b

−fgcolor=r,g,b

−setab=N (0-255; 0=black, 255=white)

−setaf=N (0-255; 0=black, 255=white)

These next options apply to mouse support.

−skipmouse
Do not enable mouse support. See MOUSE SUPPORT below for more details.

−showbuttons

This option displays clickable buttons across the bottom of the screen that execute E functions by a mouse click. The buttons appear as ascii text, eg: "... +Pg/-Pg open close/-cl pick/put ...". A screensize greater than 90 chars is recommended. This feature should be considered *experimental.*

−buttonfile=filename

This option allows one to customize which buttons to include, the order, and the text label of a button. The **−showbuttons** option needs to be in effect. The format of an entry closely matchs that of a keyboard file entry. Example:

"ins":<insmd># toggle insert mode
"+Window":<+win># toggle insert mode

See examples/e_button{1,2,3} for further examples.

−buttonfont=N (0-255; 0=black, 255=white)

This option sets the background color of buttons if the −showbuttons option is in effect. The default is the same value used for highlighting text. Some values to try are: 3, 7, 87, 136, 151, 194, and 222.

−mouseinit="initstr"

Set a custom mouse initialization string. Example: -mouseinit="\E[?1006;1002h".
The default is "\E[?1002h";

−mousestop="resetstr"

Set a custom mouse reset string. Example: -mousestop="\E[?1006;1002l".
The default is "\E[?1002l";

−noextnames

Disable the use of extended names for TERM entries in the terminfo capabilty database. Recent versions of the terminfo database contain several entries that include an XM entry that sets the mouse protocol to SGR mode. If you're using a very old terminal emulator and you find that mouse motion is not working, try using this option to disable use of XM entries. By default, E sends the mouse initialzation string of "\E[?1002h" which enables mouse position reporting while a button is pressed and dragged.

ENVIRONMENT VARIABLES

The TERM environment variable defines the terminal type. Refer to TERMINALS SUPPORTED for those terminal which *e* has terminal specific code compiled in. If there is no compiled in code for the terminal *e* will look in "/etc/termcap" for a definition of the terminal. The −dtermcap option will force *e* to use the "termcap" definition.

The environment variable EKBD defines the keyboad type if different from the TERM type. The EKBFILE environment variable points to a keyboard definition file (see the *./etc* directory in the distribution for examples) to allow the editor to use keyboard layouts that are not built in. If you're using both e19 and e20 and have added function key definitions supported in e20, set the environment variable EKBFILE_NCURSES to the e20 keyboard file definitions (see ./etc/kb_ncurses for an example).

The EDITALT environment variable will be used as the name of the alternate file. If an alternate filename is given on both the command line and with the EDITALT variable, the command line file is used.

The EPROFILE (or EPROFILE20) environment variable will be used as the name of the "startup" file, in lieu of "./.e_profile" or "~/.e_profile". The EPROFILE20 "startup" file can contain options and commands unique to e20. If both EPROFILE and EPROFILE20 are set, the latter is examined first. The command line option −profile=file wins if both are used.

The E_ETCDIR environment variable overrides the editor's built-in notion of the "etc" directory (normally */usr/local/lib/e). This is useful for installations where management will not allow* the installation of a random user's files into the system's */etc* directory. This is not generally an issue today, when desktop boxes are unique to each user.

FILES

E creates and maintains several ".e" files. They are used to keep track of the current editing session and can normally be ignored by the user. The following information is for the curious.

File .es1 contains state information on the last edit session. It is created or updated at the end of a normally terminated edit session.

File .ec1 indicates the last edit session in this directory crashed.

File .ek1 contains a history of the last edit sessions keyboard inputs. The .ek1 file is updated at regular intervals so that a mimimum of work is lost on edit or system crashes.

File .ek1b is the .ek1 file prior to a recovery replay.

The presence of any .ek1* file indicates that the last session in this directory abnormally terminated and that the next edit session will do recovery processing unless option –norecover is used. File .ek1 means abort, .ec1 means crash.

Note that if the login user is not the owner of the current directory the above files will be suffixed with the login users name. If the user has no write permission in the current directory, these files are created in the directory /tmp/etmp.

There are also a few utility files in the directory /etc/e.

PSEUDO-FILES

E maintains two pseudo-files, **#o** and **#p.** Neither of these represent actual files in the filesystem. **#o** contains all the text which has been <CLOSE>d from the file currently being edited. **#p** contains all the text which has been <PICK>ed from the file currently being edited. To make use of these pseudo-files, type **<CMD>e #o** or **<CMD>e #p** as if these were normal files. The edit window will switch to the corresponding pseudo-file, where you may examine or <PICK> text at will. Typing <ALT> (^X–A on the standard keyboard) will switch back to the file being edited.

RECENT CHANGES

E20 changes include:

- code cleanup eliminating compiler warnings; bug fixes
- now allows editing of a filename that contains embedded spaces.
  Example: CMD: e "my file" or CMD: e my\ file.
- new crash recover options, allowing playback to stop early
- revised state file (.es1.userid.host) to restore the settings and options when resuming a previous E session (ie, when E is started without a filename).
- added mouse support to: move cursor, change windows, and press/drag left button to mark a rectangular area
- added color options to highlight marked areas.
- added brace matching mode; when enabled, resting the cursor on a open/closing brace will highlight the corresponding closing/opening brace. If the matching brace is not in the current window, the column and row number of the location will be displayed in the info window. By default, the search range is ± 100 lines. The **–set brace range N** option changes the range (a value of 0 removes the limit). If the range is 0, the search range is the beginning or end of file. See the *set options* section below.
- added the **-showbuttons** option to display clickable function buttons
- added keyboard file options to map F1-12 function keys to E commands, examples: KEY_F2:<redraw>, SHIFT_F4:<+word>, CTRL_F4:<-word>; see etc/kb_ncurses for other examples. The environemnt variable EKBFILE_NCURSES may be set to a custom e20 kbfile for sites using both e19 and e20. Note that some terminal emulators use a few function keys for their own purposes, eg: gnome-terminal uses F1 for a Help menu, and F11 to toggle full screen mode.
- The containing terminal windows can now be resized, and "e" will resize its own windows in response. Also, if the current session has multiple "e" windows, with the mouse in the current window, the left and/or top border between windows may be dragged to resize the "e" windows.

E19+ changes include:

- new options **–tab, –lit, –nostrip, –nolocks**.

- better keystroke file handling (for BSD systems the keystroke file is automatically updated after 1 min. of inactivity).
- temp files: a compile option for the name to be .esN.login.hostname; and a compile option to save them after a successful recovery.
- files are no longer limited to 32767 lines (LA package revised).
- a facility to record and playback keystrokes at a later time.

E19 changes include:

- a word processing mode (WP) that currently features "word wrap" and support for a "left margin";
- a "startup" file mechanism (e.g., ~/.e_profile) to individually tailor options; also provides support for "editor scripts"
- a delete word function;
- options to control the behavior when text is typed past the right edge of the window;
- an alternate syntax for the "goto" command;
- display of tab settings on the top border, and left and right margin (width) settings on the bottom border;
- an option for invoking *E* with a previous state file;
- a new help facility;
- improved support for the "track" command;
- a new command "caps" to convert lower case to upper case;
- a new command "ccase" to reverse the case of the affected text;
- changes to *fill/just* to no longer filter out CTRL chars, and (by default) not split hyphenated words at the end of a line;
- a new "−readonly" option to prevent unintentional changes; files that are not writable cannot be modified;
- improved VT100 terminal support; and terminal drivers for the Beehive MicroBee, Televideo 910, Liberty Freedom 100, and Wyse 50.
- the sequence "<CMD><WINRIGHT>" now shifts the window far enough right to display the end of the current line (previously, this sequence would set the left edge of the window at the cursor position).
- bug fix: if an *altfile* was specified on the command line, and the environment variable EDITALT was set, the *altfile* is now used

E18 changes include:

- support for regular expression searching and replacing;
- implementation of the +WORD and −WORD functions;
- a new **set** command to individually tailor various parameters (e.g., the number of cols a <WIN RIGHT> will shift;
- support for symbolic links (the link is read and the referenced file is used);
- now runs on System III, System V, and 68000s;
- commands that specify filenames (e.g., " CMD: e *file* expands a leading "~" to the appropriate home directory;
- compiled in code for Ann Arbor Ambassadors now downloads the standard keyboard into the function keys;
- support added for Sun2 workstation.

MOUSE SUPPORT

The current (2021) rework of the editor has introduced elementary support for the mouse. The cursor may be moved arbitrarily by clicking the left mouse button in any window. If multiple windows are active in E, clicking in a non-active window will change E's focus to that window. Dragging with the left mouse button will mark rectangular areas of text. Setting the environment variable TERM=xterm-256color is recommended to make use of full color support.

If you find that dragging the mouse does not work, try setting TERM to one of these: xterm-color, xterm-1002, or xterm-x11mouse. Alternatively, the terminfo file e20/terminfo/xterm-256color-a.src can be compiled and installed with 'tic' creating the file ~/.terminfo/x/xterm-256color. (N.B. The "/x/" in this path may be replaced with an alternative directory name, .e.g. "/78/", depending on the version of "tic" you are using.) The xterm-256color-a.src file contain modified XM and kmous entries that work with older terminal emulators.

The MARK command, described in the COMMANDS section below, is used to select rectangular areas of text for use in a subsequent command. The middle mouse button can be used to start the beginning of a marked area or to cancel the marked area. A marked area is normally expanded or reduced by cursor movement (arrow keys, ± page, ± line); the area can now also be changed by clicking the left mouse button.

The ability to copy/paste text from a separate desktop window into an E window can be very useful. One method to copy/paste text into an E window is to hold down the Shift key before doing the paste. An alternate method is to temporarily disable the mouse in E by issuing the *set mouse off* command to disable the mouse in E, perform the copy/paste into E, then reenable the mouse with the command *set mouse on.* See the SET command section below. Another method is to define a function key in a keyboard file to toggle the mouse on/off. Example: KEY_F6:<mouse>.

HIGHLIGHTING

By default, areas selected using the MARK command are highlighted with black text on a light gray background. The foreground and background colors may be set by one of two command line option pairs. To choose colors using the RGB color system (red, blue, green), use options:

−*bgcolor=r,g,b,* (background) and
−*fgcolor=r,g,b* (foreground).

To choose colors using a terminal's built-in colors, use options:

−*setab=N* (background), and
−*setaf=N* (foreground).

For the RGB method, each *r, g, b* number may range from 0 to 999; higher numbers specify lighter colors. 0,0,0 specifies black; 999,999,999 specifies white. For a terminal's built-in method, *N* may range from 0 (white) to 255 (black). The colors from 1 to 254 may vary.

Since many desktop terminals support 256 colors, it is recommended that custom colors be specified using a terminal's built-in colors: −*setab* and −*setaf.* The default editor values are: −*setab=7* and −*setaf=0.* The utilities *colorhints* and *txcolor,* distributed with the editor, may give hints on selecting values for a terminal.

MACROS

The editor has elementary support for macros. This support is in the form of a "record/play" capability. As documented in the section "COMMANDS", the editor supports "record", "play", and "store" commands. The command *<cmd>record* begins recording editor commands and typed text, until a second *<cmd>record* command is given. The command *<cmd>-record* will also stop recording; the first version is provided so that it may be assigned to a key via a keyboard description file which can be used both to start and stop a macro recording. The macro that has been recorded may then be replayed by the command *<cmd>play [N]*. The optional count *N* gives the number of times the macro should be replayed; once is the default. The macro thus recorded may be assigned a name by executing *<cmd>store [name]*. The named macros may be replayed by the command *<cmd>$name [N]*.
Macros may be saved to a file. This is done via the *update macros* editor command. The file is named, by default, "~/.e_macros". This name may be overridden either by the environment variable EMACROFILE or by means of the "−macrofile=filename" command line option. This file will be read in automatically. The command line option "−nomacros" prevents this. The command *<cmd>undefine macroname* is used to remove a macro definition, which is useful prior to issuing the "<cmd>update macros" command to remove a definition from the *.e_macros* file.

COMMANDS

The following commands are available to the user by opening the command line via the <CMD> key and entering the command. The entered command is parsed and executed when the <RETURN> key is typed.

In the following discussion, pound sign ("#") introduces a comment, brackets ("[","]") indicate options, and vertical bar ("|") indicates optional choices.

There are two types of areas: a range of lines (indicated by <range>) and a marked rectangle of text (indicated by <rectangle>). Some commands will take either type of area, this is indicated by <area>. A <range> can be specified either by marking or by a number followed by "l" for lines or "p" for paragraphs;

rectangular areas must be marked. The default range is one line for most commands (fill, adjust and center default to one paragraph).

b | bye | ex | exit [nosave | quit | abort | dump]

Exit the editor. The different exit options affect whether or not the editor updates the edited files or not, and whether the ".e" state files are changed. The following table give the complete exit story.

	Saves Files	Update State	Remove Keys file	Remove Change file
exit	X	X	X	X
exit nosave	-	X	X	X
exit quit	-	-	X	X
exit abort	-	-	-	X
exit dump	-	-	-	-

blot adjust | close | erase | pick | run | box
−blot adjust | close | erase | pick | run | box
cover adjust | close | erase | pick | run | box
insert adjust | close | erase | pick | run | box
overlay adjust | close | erase | pick | run | box
underlay adjust | close | erase | pick | run | box

All of these commands take the named buffer (one of adjust, close, erase, pick, run, or box) and add or merge it into the current file.

Insert takes the named buffer, moves the existing text to make room for it, and then inserts it at the current text location. It is identical to the −pick command, except that now the other named buffers can also be inserted.

Cover places the named buffer over top of the text without moving anything.

Overlay is like cover, except that only the printing characters in the buffer clobber the original text.

Underlay is the reverse of overlay, i.e. the buffer covers only the non-printing characters of the original text.

Blot is like overlay except that positions in the file corresponding to printing characters in the buffer are erased.

−blot is like blot except that positions in the file corresponding to blanks in the buffer are erased.

Not all of the combinations listed above have been implemented.

box <rectangle>

Draws a box around the rectangle with "+", "|", and "-" characters.

caps <rectangle>

Caps changes the "target text" to all upper case characters (only the characters a-z are affected). The *mark* command is first used to define the "target text".

ccase <rectangle>

Ccase changes all upper case to lower case and vice versa. The affected area is first defined by the *mark* command.

center [<range>] width=n
fill [<range>] width=n
justify [<range>] width=n

These commands all act upon the range of text and then replace them with the modified results. Center centers the text around the column (width/2), fill places as many words as possible on each line, and justify is like fill except that it also provides a smooth right-margin by embedding blanks.

The old text is saved in the adjust buffer. As of e19, fill/just no longer filters out CTRL characters, and no longer splits hyphenated words at the end of a line. The command "CMD: set hy" will reinstate splitting.

close [<area>]
−close
Close deletes the text from the file and remembers it in a "close buffer", −close puts the close buffer back into the file at the current cursor position.

command
−command
Usually you issue commands one at a time by typing the <CMD> key, the command, and then <RETURN>. The "command" command places you in a mode where you can keep typing commands without hitting the <CMD> key each time. To return to normal edit mode, issue the −command command.

delete
Tells the editor to delete the current file on exit.

dword
−dword
Delete word. If the cursor is "on" a word, that word is deleted, othwise the next word is deleted. The deleted word can be restored by −dword.

e | edit [<filename>]
If no filename is specified, the alternate file (if any) is brought up in the current window. Otherwise the specified file is displayed. If *filename* begins with "~" or "~/", the appropriate home directory is prepended.

erase [<area>]
−erase
Erase replaces the area by blanks and puts the contents into an "erase" buffer; −erase inserts the contents of this buffer at the current cursor position.

feed [<range>] command-string
A Unix shell is started with the command-string, and the text in <range> is supplied to the shell as the standard input. Any results of the command are inserted before the first line in the range.

goto [b | line−number | e | rb | re]
Goto is a quick way of moving around a file. Goto 'b' moves to the beginning of the file, 'e' moves to the end. A line-number goes to that line of the file; 'rb' and 're' move to the beginning or end of a marked range (see the range command). A short-cut to goto line-number *n* is "CMD: *n*<RETURN>".

help
Help will clear the screen and (for some terminal types) show a map of the keyboard layout of the standard keyboard, showing the command tied to each keycap. The user may then type any control character, and a short paragraph documenting the associated command will be shown. Note that the keyboard shown does not reflect any optional keyboard file in use, for example by means of the EKBFILE environment variable.

join | −split
split | −join
Join (or −split) combines two lines, split (or −join) breaks a line in two.

?macros
Lists macros which have been saved by the *store* command.

?macros [X]
Shows the keystrokes which have been stored in macro X by the *store* command.

name <newfilename>
Tells the editor to rename the current file to newfilename when it exits.

open [<area>]
Move existing text to make room for <area> worth of blank lines (or a rectangle of blanks).

pick [<area>]
−pick
Pick copies the <area> to a pick buffer, −pick inserts the pick buffer at the current cursor position.

play [N]
Begin playback of a recording, once or *N* times.

range [<range>]
−range
?range
The range command is another method of limiting the range of some commands (notably the replace, fill, justify and center commands). −range turns it off, ?range tells you the line numbers of the range area.

record
−record
Begin (or end) saving keystrokes for later playback using the *play* command. In recording mode, the only mouse clicks saved in a recording are button clicks while in **−showbuttons** mode. A mouse click in a window area is ignored while recording.

redraw | red
The redraw command is used to erase and redraw the editor windows in case something has happened to it (for example, line noise on dialup lines, or messages from the operator).

replace [<range>] [<option>] /search-string/replace-string/
−replace [<range>] [<option>] /search-string/replace-string/
The replace command searches forward over the range replacing the search-string with the replace-string; −replace searches backwards doing the same thing. The '/' delimiter can be replaced by other symbols if the search or replace strings contain a slash. Any non-control, non-alphanumeric character can be used as the delimiter.

Two options can also be specified, they are *show* and *interactive.* The *show* option allows the user to see the replacements as they occur. The *interactive* option additionally allows the user control over whether or not to make each replacement by displaying the search-string and allowing the user to hit the <REPL> key to do replacement, or the <±SRCH> key to skip the replacement.

regexp
−regexp
The "regexp" command enters regular expression mode (for searching and replacing); "−regexp" exits regular expression mode. While in regular expression mode, "RE" is displayed at the bottom of the screen.

run [<range>] command-string
The run command is similar to the feed command, except that the marked text is deleted and replaced by the results of running the command-string.

save <newfilename>
Immediately write a copy of the file to newfilename.

set *option*
?set
used to set various options:

{}	on|off, toggles brace matching mode (shortcut)
bracematch	on|off, toggles brace matching on/off
brace range *n*	sets brace matching range to *n* lines (default: 100)

brace ?	shows the brace range value (default: 100 lines) setting the limit to 0 (or "off") extends the search to the end of file.
brace coding	on|off, sets brace matching to "coding" mode where a search stops at the end of the current function(); assumes that a curly brace in column one ends a function() definition.
+line *n*	sets the <+line> key to *n* lines
−line *n*	sets the <−line> key to *n* lines
line *n*	sets the <+line> and <−line> keys to *n* lines
+page *n*	sets the <+page> key to *n* screens
−page *n*	sets the <−page> key to *n* screens
page *n*	sets the <−page> and <+page> keys to *n* screens
?	displays the values of several options (same as the "?set" command)
bell	enable the bell
nofilldot	a fill cmd with no parameters will stop at a line beginning with a period (this is the new default)
filldot	fill will not stop at lines beginning with a period (as before)
nobell	disable the bell
highlight	set mode to: on|off|color|bold|rev
hy	enables splitting of hyphenated words by fill/just
nohy	disables splitting of hyphenated words by fill/just (default)
left *n*	sets the <window left> key to *n* cols
lmar *n*	sets the left margin to *n*
mouse	on|off enable/disable mouse
right *n*	sets the <window right> key to *n* cols
rmar *n*	sets the right margin (linewidth) to *n*
window *n*	sets the <window left> and <window right> keys to *n* cols
width *n*	sets the linewidth to *n* cols (for fill, etc.)
wordelim *mode*	sets the action of the <+word> and <−word> keys. If mode is *whitespace* (the default), words are delimited by blanks and newlines, and the cursor advances to the first character following the delimiter. If mode is *alphanum*, words are delimited by all special characters in addition to blanks and newlines, and the cursor advances to the first alphanumeric character following the delimiter.

The "line", "page", and "window" options may be individually set in different windows.

stop

Use 'job control' to suspend the editor. Returns control to the shell, resume with the 'fg' command. Works under most modern operating systems.

store X

Save the current recording in a macro named X for later playback by "<cmd> $X [N]", which inserts the contents of macro X at the current cursor position, once or *N* times. Up to ten macros may be stored for playback.

tab [column ...]
−tab [column ...]

Tab sets tabs in the specified columns; −tab removes tabs in specified columns.

tabs n
−tabs

Tabs sets tabs every nth column, and −tabs removes all tabs.

tabfile tabfilename
−tabfile tabfilename

Tabfile sets tabs (−tabfile clears tabs) at every column listed in the tabfilename.

tag [keyword | + | -]

The tag command is useful only for editing computer source code, not general text. It is used to find the definition of a function or variable name, specified either on the cmd line or whose name is currently under the cursor. It uses a *tags* file generated by the *ctags* utility. This file must be in the current directory in which the editor is run. If no argument is given, the editor will open the file containing the function definition and place the cursor on the name of the function in the definition. If the *tags* file contains multiple matching tags for the target, the command "tag +" will advance to the next definition, and "tag -" will back up to the previous definition. If there is a second window open in the editor, that window will be used to display the function or variable definition, leaving the original window unchanged. This allows the programmer to use the tag facility to look up a definition without changing the current coding position.

Currently, only C language tags are known to work.

track
−track

The track command is used to scroll the main and alternate files together. This is normally used to visually compare the main file and the alternate file by rapidly changing between them and watching what changes on the screen (similar to blink-comparators as used in astronomy). The −track command turns off the tracking mode. As of e19, the TRACK flag has been made part of the state of each window, so that the TRACK mode can be reestablished following an interruption, and the TRACK command has been changed so that it toggles the TRACK mode, in the same way as does INSERT.

undefine macroname

Removes macroname. This command is useful prior to issuing the "update macros" command to remove a macro definition from the *.e_macros* file.

update | −update [inplace | −inplace | −readonly | macros]

The update command tells the editor what to do on exit with the current file. The update command can be used to specify whether or not to break links, the −update command causes any changes to the current file to be ignored. The *−readonly* option is used to enable modifications to files which (1) you own but lack write permission on, or, (2) to override a "−readonly" invocation argument. The *macros* option causes any defined macros to be saved to the macros file as specified in the section MACROS.

w | window [filename]
−w | −window

The window command will create a new window at the current cursor position (as long as the cursor is along the top or left margins). If a filename is specified, that file is displayed in the new window, otherwise the current file is used. The −window command deletes the current window.

wp
−wp

The *wp* command enters word procesing mode, *−wp* exits. A WP is displayed at the bottom of the screen in word processing mode. WP mode enables power typing or "word wrap", where any text that is entered past the right margin (see "set rmar" above) is automatically placed on the following line at the left margin (see "set lmar" above). WP mode also moves the cursor to the left margin in response to a <RETURN>. If WP mode is entered while a marked area is in effect, the boundaries of the marked area become the left and right margin settings. N.B.: Every line break introduced by WP will cause the line involved to be added to the **#o** pseudo-file.

FUNCTION KEYS

E supports a set of *functions* that are normally executed by hitting one of the terminal functions keys, or by typing a specific control- or escape sequence. Details of what *E-functions* are emitted for a specific terminal's function key layout are hard to document generally, since each site usually tailor's this to their liking. Your system administrator should be able to provide a mapping. Also, see the following sections *Standard Editior Keyboard* and *Terminals/Keyboards Supported*. Note that some *E-functions* require first pressing the <CMD> key followed by another <KEY>. Refer to the previous section "COMMANDS" for a discussion of the following functions: <CLOSE>, <OPEN>, <PICK>, <ERASE>, and <MARK> which are

normally implemented on function keys. The sequences <CMD><CLOSE>, <CMD><PICK> and <CMD><ERASE> are documented above as −*close,* −*pick,* and −*erase*.

<←> | <↑> | <↓> | <→>
: The arrow keys move one space (or line) in the indicated direction. An <↑> on the top line of the window (or <↓> on the bottom line) scrolls the window down (or up) one line. A <←> at the left edge of a window does nothing. A <→> at the right edge of a window does nothing if option −*stick* is set, otherwise the window is shifted right (16 spaces by default).

<+LINE>, <+LINE>
: scrolls window down/up about 1/4 the window size.

<CMD><+LINE>, <CMD><−LINE>
: makes the current cursor line the top (or bottom) line of the window.

<+PAGE>, <−PAGE>
: scrolls window down/up a full "page".

<CMD><+PAGE>, <CMD><−PAGE>
: moves to the end/beginning of the file.

<+SRCH>, <−SRCH>
: searches in the indicated direction for the next occurrence of the "search string". The *search string* is set by <CMD> search string <±SRCH>.

<CMD><+SRCH>, <CMD><−SRCH>
: sets the "search string" to the current cursor word, and performs a search in the indicated direction.

<+TAB>, <−TAB>
: tabs in indicated direction.

<+WORD>, <−WORD>
: moves to next/previous word.

<ALT> switch to alternate file, also "<CMD>e<RETURN>".

<BS> backspace, moves left and erases previous character. <CMD><BS> deletes text from the beginning of the line up to the current cursor character. In *insert mode*, the rest of the line is moved left.

<CHWIN> change to next window.

<CTRLCHAR>
: used to enter non-ascii characters (\000−\037) in text.

<DELCHAR>
: deletes cursor character.

<CMD><DELCHAR>
: deletes text from the cursor to the end of the line.

<HOME> moves to upper left corner of window.

<INSERT> toggles *insert* mode.

<INT> Interrupts the operation of various functions: e.g., searches.

<MARK> used to define a range of lines or rectangular area of text. A simple <MARK> will start by defining one *line*. Moving the cursor up/down will extend the defined range of *lines* (as will other keys like <+LINE> <+PAGE>). When full lines are *marked*, the message "MARK *n*" is displayed at the bottom of the screen. Rectangular areas of text are marked by first marking one or more lines, then moving the cursor left or right. When a rectangular area is marked, the message "MARK *lines* x *cols*" is displayed.

<CMD><MARK>
cancels any mark in affect.

<REPL> Executes the "replacement" while doing an interactive replace command.

<RETURN>
Positions cursor at column 1 on the next line. A <RETURN> on the bottom line of a window is equivalent to a <+LINE>. In WP mode, a <RETURN> positions the cursor at the current *left margin* on the next line, which may automatically shift the window left.

<WINLEFT>, <WINRIGHT>
Shifts the window left/right (16 spaces by default).

<CMD><WINLEFT>
Shifts the window all the way back to column 1.

<CMD><WINRIGHT>
Shifts the window as far right as necessary to display the end of the line.

STANDARD EDITOR KEYBOARD

There is now a "standard" E keyboard, that is designed to be usable on all video display terminals. Either say "e −keyboard=standard" or "setenv EKBD standard" before running E to select this keyboard. This is also the keyboard that E will use if there is no specific knowledge of your type of terminal compiled into the editor.

This keyboard layout is designed to be used on terminals with no function keys. "^H" means control−H, and "^X−^U" means control−X followed by control−U. The ALT entry gives you a choice of "^/" or "^_" because one or the other or both will not work on some terminals. ("_" is correct ASCII.)

E STANDARD KEYBOARD

(extend)	^X	DOWN	^J
+LINE	^F	ERASE	^X-^E or ^^
+PAGE	^R	HOME	^G
+SRCH	^Y	INSERT	^Z or ESC or ^[
+TAB	^I	INT	^\
+WORD	^N	JOIN	^X-^J
-LINE	^D	LEFT	^H
-PAGE	^E	MARK	^U
-SRCH	^T	OPEN	^O
-TAB	^X-^U	PICK	^P
-WORD	^B	REPL	^X-^R or ^]
ALT	^X-^A or ^_ or ^/	RETURN	^M
BSP	^C	RIGHT	^L
CHG WIN	^X-^W	SPLIT	^X-^B
CLOSE	^V	TABS	^X-^T
CMD	^A or NULL or ^@	UP	^K
CTRLCHAR	^X-^C	WIN LEFT	^X-^H
DELCHAR	^W	WIN RIGHT	^X-^L

^@	CMD	^L	RIGHT	^X	(extend)	^X-^H	WIN LEFT
^A	CMD	^M	RETURN	^Y	+SRCH	^X-^J	JOIN
^B	-WORD	^N	+WORD	^Z	INSMODE	^X-^L	WIN RIGHT
^C	LEFT	^O	OPEN	^[	INSERT	^X-^N	DEL WORD
^D	-LINE	^P	PICK	^\	INT	^X-^P	PLAY
^E	-PAGE	^Q	(not used)	^]	REPL	^X-^R	RECORD
^F	+LINE	^R	+PAGE	^^	ERASE	^X-^B	SPLIT
^G	HOME	^S	(not used)	^_	ALT	^X-^T	TABS
^H	BSP	^T	-SRCH	^/	ALT	^X-^U	-TAB
^I	+TAB	^U	MARK	^X-^A	ALT	^X-^W	CHG WIN
^J	DOWN	^V	CLOSE	^X-^C	CTRLCHAR		
^K	UP	^W	DELCHAR	^X-^E	ERASE		

TERMINALS/KEYBOARDS SUPPORTED

This is installation specific information with the actual terminal types defined varying across the host cpu's versions of the editor. Now that the editor can be driven from termcap and the standard keyboard, many sites will wish to delete specific terminal type support to save memory. (N.b: These days, the amount of memory thus saved is several orders of magnitude less than anybody would conceivably be worried about). Code is provided with E for the following terminals and may be compiled into your version.

Also, note that the EKBFILE environment variable may be used to redefine the keyboard at will on an individual user basis.

DESCRIPTION	TERMINAL TYPE
Ann Arbor S001901/2	aa, aa0, annarbor, default
New ann arbor (Model Q2878)	aa1

Ann Arbor Ambassador	aaa, ambas, ambassador, aaa-N
and Ann Arbor XLs	(N is one of 18,20,22,24,26,28, 30,36,40,48 or 60)
Beehive MicroBee	microb
Termcap terminal	[from /etc/termcap]
INTERACTIVE Systems Intext2	intext2, T_2intext
Lear Siegler ADM3a	3a, adm3a, la
Liberty Freedom 100	fr10, free100
Heathkit H19 & H89	h19, k1
INTERACTIVE Systems Intext	in, intext
Perkin Elmer 1251	po
Standard keyboard	standard
DEC VT100 (needs termcap)	vt100, vt100w (wide)
Lear Siegler ADM31	31, adm31, l1
Lear Siegler ADM42	adm42, l4
Concept 100	c100, co, concept, concept100
Datamedia DM4000	dm4000
Wyse 50	wyse50, wyse50w (wide)

PLATFORMS SUPPORTED

This version of the editor has been ported to, and tested on, Linux systems (specifically CentOS) and Mac systems (specifically Mojave and Catalina). It is not a "clickable" app; it must be run in a terminal window.

The Linux version has been run under the Linux subsystem (specifically Ubuntu) on Windows 10, but has not been extensively tested there. Or, really, tested at all.

To build the distribution on a given platform, read the README.md file in the GitHub distribution. In brief, look in the top-level Makefile file to find the target for your platform ("sys5" for Linux, "bsd" for Mac). Then, read the e19/Makefile file to see what reconfiguration must be done to build your target. "e" was built long, long before the *config* utility was developed.

STARTUP FILE

As a means of individually tailoring various options on startup, E will run commands specified in a "profile" file. The order in which E decides which "profile" file to run, is: command line option −profile=*file*, environment variable EPROFILE20, then environment variable EPROFILE, a "safe" *./.e_profile*, *~/.e_profile20*, and finally, *~/.e_profile*. ("*./.e_profile*" is deemed to be safe if you are not the super-user, you own the file, and it's not writable by everyone else.) The EPROFILE20 environment variable and *~/.e_profile20* file are searched first in case the original RAND editor branch is also in use. The original editor does not support the new features of the current version. These new features, if present in a profile file, would cause the old version of the editor to exit with an error. Therefore, for convenience, these new features may be placed in a profile file that the old version of the editor will not search for. If the first line of the profile file begins with the keyword "options:", then the rest of the line is treated as if typed as initial arguments to E. The options must currently all be on one line.

Profile format notes:

- blank lines and lines beginning with # are ignored
- E function keys are denoted by "<keyword>"; use "\<" to insert a "<" in text; "<#keyword>" may be used to repeat *keyword* #-times
- keywords:

<+line>	<caps>	<int>	<right>*
<+page>	<ccase>	<join>	<split>
<+sch>	<cchar>	<left>*	<srtab>
<+tab>	<chwin>	<mark>	<tab>
<+word>	<close>	<mouse>	<up>*
<−line>	<cmd>	<open>	<wleft>
<−mark>	<dchar>	<pick>	<wp>
<−page>	<down>*	<play>	<wright>

<−sch>	<dword>	<record>	<null>
<−tab>	<edit>	<redraw>	<undef>
<−word>	<erase>	<regex>	
<bksp>	<home>	<replace>	
<brace>	<insmd>	<ret>	

(*=cursor motion)

- essentially anything that can be typed at E can be inserted in the ".e_profile". Thus, application specific editor scripts can be created.
- note that various CMD options end with a <ret>, others don't; e.g., "<cmd>tabs 4<ret>", but "<cmd><+page>".
- E looks for a "profile" file and examines the "options:" line *before* processing any command line arguments, thus command line options override any options specified in the "options:" line. *Caveat*: if the −profile=file argument is used, any "options:" in a profile file will have already been processed. This is a "cart before the horse" problem....
- any format errors will terminate reading of the profile file, and continue the E session;
- example:

```
options: −regex −stick
<cmd>tabs 4<ret>
<cmd>set window 40<ret>
```

There is an "examples/" directory at the top level of the distribution which contains sample *.e_profile* and *.e_profile20* files.

BUGS

By far the worst problem with E is its treatment of tabs. E is a "whitespace" editor, all whitespace is considered equal. On any line that E modifies, it will compress multiple initial blanks to tabs, strip trailing blanks, and convert any embedded tabs to blanks. This is acceptable for most editing, but some programs expect real tab characters as input, for example tbl(1) and some nroff commands. However, the new options **−lit, −nostrip, −tabs**, and **−blanks** alleviate most of these problems.

The "Terminal" application on the Mac has problems with use of the *-bgcolor* and *-fgcolor* arguments. The application does its own processing, and prevents the specified colors from being used. The *-setab* and *-setaf* arguments work correctly. An alternative is to use another terminal application on the Mac. *iTerm2* is known to work.

Resizing terminal windows in the parent operating system is not supported.

SEE ALSO

N-2056-1: SELF-TEACHING GUIDE TO RAND'S TEXT PROCESSOR, 1986, The Rand Corporation

N-2239-1: THE RAND EDITOR, e, Version 19, 1986, The Rand Corporation.

These are available from *https://rand.org.*

Two other documents worthy of note are R-2000-ARPA, *A Guide to NED: A New On-Line Computer Editor, 1977,* and R-2176-ARPA,

SPONSOR

The Rand Corporation.

NAME

e - the RAND editor, E

SYNOPSIS

e [options] [mainfile [altfile]]

DESCRIPTION

The Rand editor *E* is a full screen editor which treats a text file as a two-dimensional surface with text on it. Among its more unusual features are manipulating rectangular areas of text, editing several files at once, and displaying text in one or more windows. While it is not as powerful as *Emacs,* or as widely used as *Vi,* most people at Rand preferred its easy to learn commands and used it to edit both programs and text, during the period ranging from roughly 1973 to 1995. *E* is a complete rewrite of the Rand *NED* editor of many years back. The current version (E19+) has been improved to take advantage of terminals that have the ability to do local insert/delete line, and it can now use *termcap* or *terminfo* so that most terminals with direct cursor addressing can be used.

Please note that this manual entry does not try to be either a tutorial or a full reference manual for the editor. Rand users should initially learn *E* by reading the RAND document *Self Teaching Guide to RAND's Text Processor, 1986,* available from *https://rand.org* as N-2056-1. Report N-2239-1, *The RAND Editor e: Version 19, 1986,* is a full reference manual for the editor, and is mostly correct for this version. Two other useful reports are R-2000-ARPA, *A Guide to NED: A New On-Line Computer Editor, 1977* and R-2176-ARPA, *The CRT Text Editor NED -- Introduction and Reference Manual, 1977.*

The *mainfile* on the command line is the file which is initially displayed in the editor. If the file doesn't exist, the user will be asked if it should be created. If in addition, an *altfile* is specified on the command line, it will be read in as the initial alternate file. If no filenames are present on the command line, *e* will attempt to continue from your previous editing session. To do this it saves "state" in a file called '.es1' in your current working directory. Before invoking the editor, be sure and specify your terminal type by setting the TERM environment variable.

Refer to section RECENT CHANGES for a list of new changes to E.

OPTIONS

The currently available command line options are:

-help

This causes a display of the currently available options and their values along with the version number of the editor.

-bullets or **-nobullets**

Options -bullets and -nobullets forces border bullets on or off regardless of line speed or terminal type. Line speeds of 1200 baud and slower and some terminal types set border bullets off in the absence of these options. "Border bullets" does not refer to the characters making up the border of the editing window in "e". They refer to four characters placed at the top, bottom, left and right of the current window indicating the current postion of the cursor. They are chiefly useful on terminals or terminal emulators which do not have a blinking cursor, but some people may prefer them as an added indication of the cursor position. If the terminal supports highlighting, they will be highlighted.

-create

If the file named on the command line does not exist, the editor will ordinarily ask if you want to create it. This option causes the editor to create the file immediately, without asking.

-newfilemode=mode

If the editor creates a new file, it will be created with mode "mode". This is of the typical UNIX octal form, e.g. "0644".

-debug=filename

The -debug option causes the editor to write debugging information to the file "filename".

-inplace

Option -inplace preserves file links except for singly-linked files. From within the editor, link preservation can be turned on and off, even for singly-linked files, by the update command.

-terminal=termtype

Option -terminal=termtype identifies the terminal type explicitly. In the absence of the -terminal option the terminal type is retrieved from the TERM environment variable.

If there is no compiled-in code for the terminal, the termcap entry for the terminal type will be used and the "standard" keyboard will be assigned. Refer to TERMINALS SUPPORTED section of this document for terminals that can be supported by compiled-in code (varies with installation).

-keyboard=termtype

-dkeyboard

Option -keyboard=termtype sets the keyboard type to "termtype". It overrides any "-terminal=" argument and also any EKBD environment variable. Option -dkeyboard sets the keyboard type to "standard". This is a shortcut for "-keyboard=standard".

-dtermcap

Option -dtermcap forces *E* to use the termcap entry for the terminal identified by the TERM variable or -terminal option, disregarding any applicable compiled-in code.

-notracks

Option -notracks will allow you to edit a file without using or disturbing the work files (keys, state, and change files) from your previous session.

-replay=filename

Option -replay=filename allows you to replay with a specified keys file, where "filename" represents that keys file. Use -silent when the replay display is not desired. The message "***replay completed" or "***replay aborted" signals the end of the replay. The <INT> key can be used to stop the replay short of completion. Files .ec1* and .ek1* must not be present if the replay is to work.

-norecover

Option -norecover will open the edit session as defined by the .es1 state file, ignoring any recovery processing of a prior crashed or aborted session.

-regexp

Option -regexp enables regular expression pattern matching for searching and replacing. This mode is also enabled by the "re[gexp]" command. In regular expression mode, "RE" is displayed at the bottom of the screen. The command "-re[gexp]" returns to non-regular expression mode. The regular expressions recognized are essentially the same as that for *ed*.

-state=filename

Option -state=filename causes the initial state of the editor to be read from the indicated "state" file. The state file from a previous *E* session (with alternate files, windows, cursor positioning, etc.) may be saved by exiting, and renaming the state file (".es1").

-[no]stick

Option -stick sets the behavior at the right margin to stop in response to text or a <RIGHT ARROW>. -nostick is the default, and will automatically shift the window right in response to text or the <RIGHT ARROW> key.

-readonly

Option -readonly will prohibit any changes to the file(s).

-noreadonly

Option -noreadonly will disable the (new e19) mechanism preventing changes to files lacking

write permission, that you could previously change because you had write permission in the directory.

-noprofile

Option -noprofile will prevent reading of the ".e_profile". (See *caveat* under STARTUP FILE.

-profile=file

Specifies the profile to be used in lieu of the default "./.e_profile" or "~/.e_profile". See STARTUP FILE in a later section.

-blanks

Expands tabs to blanks. The keyletters BL are be displayed at the bottom of the screen.

-literal

Displays tabs in ^I notation. The keyletters LIT are displayed at the bottom of the screen.

-tabs

Don’t convert tabs to blanks. The keyletters TAB are displayed at the bottom of the screen.

-nostrip

Do not remove blanks at the end of lines. The keyletters NOS are displayed at the bottom of the screen.

-nolocks

Turns off file locking if the editor was compiled with the option FILELOCKING.

-nomacros

Prevents reading a ".e_macros" file if one exists. See MACROS in a later section.

-search=string

Sets the initial search string.

NEW OPTIONS IN E20

-nowinshift

In previous editions of the editor, creating a new window on the current file by typing "CMD: window" would obscure one line of text in the file with the top border of the new window. This avoided the necessity of repainting the text in the new window by shifting it down, which was desirable in the days of slow serial lines. The current version of the editor shifts the new window down by one line, to allow all of the text in the file to be visible in one window or the other. Option -nowinshift causes the editor to revert to the old behavior.

-noresize

The editor now supports resizing of the terminal window in which it's running. It will resize its own windows to match. There are a few situations in which this adaptive behavior is undesirable. This option prevents the editor from resizing its own windows when the terminal window in which it's running is resized.

These next four options control the foreground and background colors used to highlight the areas selected by the MARK command. If you have 256 colors available, some values to try for the **-setab=N** option are: 3, 7, 87, 136, 151, 194, and 222.

See MOUSE SUPPORT for details.

-bgcolor=r,g,b

-fgcolor=r,g,b

-setab=N (0-255; 0=black, 255=white)

-setaf=N (0-255; 0=black, 255=white)

These next options apply to mouse support.

-skipmouse

Do not enable mouse support. See MOUSE SUPPORT below for more details.

-showbuttons

This option displays clickable buttons across the bottom of the screen that execute E functions by a mouse click. The buttons appear as ascii text, eg: "... +Pg/-Pg open close/-cl pick/put ...". A screensize greater than 90 chars is recommended. This feature should be considered *experimental.*

-buttonfile=filename

This option allows one to customize which buttons to include, the order, and the text label of a button. The **-showbuttons** option needs to be in effect. The format of an entry closely matchs that of a keyboard file entry. Example:

"ins":<insmd># toggle insert mode

"+Window":<+win># toggle insert mode

See examples/e_button{1,2,3} for further examples.

-buttonfont=N (0-255; 0=black, 255=white)

This option sets the background color of buttons if the -showbuttons option is in effect. The default is the same value used for highlighting text. Some values to try are: 3, 7, 87, 136, 151, 194, and 222.

-mouseinit="initstr"

Set a custom mouse initialization string. Example: -mouseinit="\E[?1006;1002h".
The default is "\E[?1002h";

-mousestop="resetstr"

Set a custom mouse reset string. Example: -mousestop="\E[?1006;1002l".
The default is "\E[?1002l";

-noextnames

Disable the use of extended names for TERM entries in the terminfo capabilty database. Recent versions of the terminfo database contain several entries that include an XM entry that sets the mouse protocol to SGR mode. If you're using a very old terminal emulator and you find that mouse motion is not working, try using this option to disable use of XM entries. By default, E sends the mouse initialzation string of "\E[?1002h" which enables mouse position reporting while a button is pressed and dragged.

ENVIRONMENT VARIABLES

The TERM environment variable defines the terminal type. Refer to TERMINALS SUPPORTED for those terminal which *e* has terminal specific code compiled in. If there is no compiled in code for the terminal *e* will look in "/etc/termcap" for a definition of the terminal. The -dtermcap option will force *e* to use the "termcap" definition.

The environment variable EKBD defines the keyboad type if different from the TERM type. The EKBFILE environment variable points to a keyboard definition file (see the *./etc* directory in the distribution for examples) to allow the editor to use keyboard layouts that are not built in. If you're using both e19 and e20 and have added function key definitions supported in e20, set the environment variable EKBFILE_NCURSES to the e20 keyboard file definitions (see ./etc/kb_ncurses for an example).

The EDITALT environment variable will be used as the name of the alternate file. If an alternate filename is given on both the command line and with the EDITALT variable, the command line file is used.

The EPROFILE (or EPROFILE20) environment variable will be used as the name of the "startup" file, in lieu of "./.e_profile" or "~/.e_profile". The EPROFILE20 "startup" file can contain options and commands unique to e20. If both EPROFILE and EPROFILE20 are set, the latter is examined first.

The command line option -profile=file wins if both are used.

The E_ETCDIR environment variable overrides the editor's built-in notion of the "etc" directory (normally */usr/local/lib/e*). This is useful for installations where management will not allow the installation of a random user's files into the system's */etc* directory. This is not generally an issue today, when desktop boxes are unique to each user.

FILES

E creates and maintains several ".e" files. They are used to keep track of the current editing session and can normally be ignored by the user. The following information is for the curious.

File .es1 contains state information on the last edit session. It is created or updated at the end of a normally terminated edit session.

File .ec1 indicates the last edit session in this directory crashed.

File .ek1 contains a history of the last edit sessions keyboard inputs. The .ek1 file is updated at regular intervals so that a mimimum of work is lost on edit or system crashes.

File .ek1b is the .ek1 file prior to a recovery replay.

The presence of any .ek1* file indicates that the last session in this directory abnormally terminated and that the next edit session will do recovery processing unless option -norecover is used. File .ek1 means abort, .ec1 means crash.

Note that if the login user is not the owner of the current directory the above files will be suffixed with the login users name. If the user has no write permission in the current directory, these files are created in the directory /tmp/etmp.

There are also a few utility files in the directory /etc/e.

PSEUDO-FILES

E maintains two pseudo-files, **#o** and **#p.** Neither of these represent actual files in the filesystem. **#o** contains all the text which has been <CLOSE>d from the file currently being edited. **#p** contains all the text which has been <PICK>ed from the file currently being edited. To make use of these pseudo-files, type **<CMD>e #o** or **<CMD>e #p** as if these were normal files. The edit window will switch to the corresponding pseudo-file, where you may examine or <PICK> text at will. Typing <ALT> (^X-A on the standard keyboard) will switch back to the file being edited.

RECENT CHANGES

E20 changes include:

- code cleanup eliminating compiler warnings; bug fixes
- now allows editing of a filename that contains embedded spaces.

Example: CMD: e "my file" or CMD: e my\ file.

- new crash recover options, allowing playback to stop early
- revised state file (.es1.userid.host) to restore the settings and options when resuming a previous E session (ie, when E is started without a filename).
- added mouse support to: move cursor, change windows, and press/drag left button to mark a rectangular area
- added color options to highlight marked areas.
- added brace matching mode; when enabled, resting the cursor on a open/closing brace will highlight the corresponding closing/opening brace. If the matching brace is not in the current window, the column and row number of the location will be displayed in the info window. By default, the search range is +- 100 lines. The **-set brace range N** option changes the range (a value of 0 removes the limit). If the range is 0, the search range is the beginning or end of file. See the *set options* section below.
- added the **-showbuttons** option to display clickable function buttons
- added keyboard file options to map F1-12 function keys to E commands, examples: KEY_F2:<redraw>, SHIFT_F4:<+word>, CTRL_F4:<-word>; see etc/kb_ncurses for other examples. The environemnt variable EKBFILE_NCURSES may be set to a custom e20 kbfile for sites using both e19 and e20. Note that some terminal emulators use a few function keys for their own purposes, eg: gnome-terminal uses F1 for a Help menu, and F11 to toggle full screen mode.
- The containing terminal windows can now be resized, and "e" will resize its own windows in response. Also, if the current session has multiple "e" windows, with the mouse in the current window, the left and/or top border between windows may be dragged to resize the "e" windows.

E19+ changes include:

- new options **-tab, -lit, -nostrip, -nolocks**.
- better keystroke file handling (for BSD systems the keystroke file is automatically updated after 1 min. of inactivity).
- temp files: a compile option for the name to be .esN.login.hostname; and a compile option to save them after a successful recovery.
- files are no longer limited to 32767 lines (LA package revised).
- a facility to record and playback keystrokes at a later time.

E19 changes include:

- a word processing mode (WP) that currently features "word wrap" and support for a "left margin";
- a "startup" file mechanism (e.g., ~/.e_profile) to individually tailor options; also provides support for "editor scripts"
- a delete word function;
- options to control the behavior when text is typed past the right edge of the window;

- an alternate syntax for the "goto" command;
- display of tab settings on the top border, and left and right margin (width) settings on the bottom border;
- an option for invoking *E* with a previous state file;
- a new help facility;
- improved support for the "track" command;
- a new command "caps" to convert lower case to upper case;
- a new command "ccase" to reverse the case of the affected text;
- changes to *fill/just* to no longer filter out CTRL chars, and (by default) not split hyphenated words at the end of a line;
- a new "-readonly" option to prevent unintentional changes; files that are not writable cannot be modified;
- improved VT100 terminal support; and terminal drivers for the Beehive MicroBee, Televideo 910, Liberty Freedom 100, and Wyse 50.
- the sequence "<CMD><WINRIGHT>" now shifts the window far enough right to display the end of the current line (previously, this sequence would set the left edge of the window at the cursor position).
- bug fix: if an *altfile* was specified on the command line, and the environment variable EDITALT was set, the *altfile* is now used

E18 changes include:

- support for regular expression searching and replacing;
- implementation of the +WORD and -WORD functions;
- a new **set** command to individually tailor various parameters (e.g., the number of cols a <WIN RIGHT> will shift;
- support for symbolic links (the link is read and the referenced file is used);
- now runs on System III, System V, and 68000s;
- commands that specify filenames (e.g., " CMD: e *file* expands a leading "~" to the appropriate home directory;
- compiled in code for Ann Arbor Ambassadors now downloads the standard keyboard into the function keys;
- support added for Sun2 workstation.

MOUSE SUPPORT

The current (2021) rework of the editor has introduced elementary support for the mouse. The cursor may be moved arbitrarily by clicking the left mouse button in any window. If multiple windows are active in E, clicking in a non-active window will change E’s focus to that window. Dragging with the left mouse button will mark rectangular areas of text. Setting the environment variable TERM=xterm-256color is recommended to make use of full color support.

If you find that dragging the mouse does not work, try setting TERM to one of these: xterm-color, xterm-1002, or xterm-x11mouse. Alternatively, the terminfo file e20/terminfo/xterm-256color-a.src can be compiled and installed with ’tic’ creating the file ~/.terminfo/x/xterm-256color. (N.B. The "/x/"

in this path may be replaced with an alternative directory name, .e.g. "/78/", depending on the version of "tic" you are using.) The xterm-256color-a.src file contain modified XM and kmous entries that work with older terminal emulators.

The MARK command, described in the COMMANDS section below, is used to select rectangular areas of text for use in a subsequent command. The middle mouse button can be used to start the beginning of a marked area or to cancel the marked area. A marked area is normally expanded or reduced by cursor movement (arrow keys, +-page, +-line); the area can now also be changed by clicking the left mouse button.

The ability to copy/paste text from a separate desktop window into an E window can be very useful. One method to copy/paste text into an E window is to hold down the Shift key before doing the paste. An alternate method is to temporarily disable the mouse in E by issuing the *set mouse off* command to disable the mouse in E, perform the copy/paste into E, then reenable the mouse with the command *set mouse on.* See the SET command section below. Another method is to define a function key in a keyboard file to toggle the mouse on/off. Example: KEY_F6:<mouse>.

HIGHLIGHTING

By default, areas selected using the MARK command are highlighted with black text on a light gray background. The foreground and background colors may be set by one of two command line option pairs. To choose colors using the RGB color system (red, blue, green), use options:

-bgcolor=r,g,b, (background) and
-fgcolor=r,g,b (foreground).

To choose colors using a terminal's built-in colors, use options:

-setab=N (background), and
-setaf=N (foreground).

For the RGB method, each *r, g, b* number may range from 0 to 999; higher numbers specify lighter colors. 0,0,0 specifies black; 999,999,999 specifies white. For a terminal's built-in method, *N* may range from 0 (white) to 255 (black). The colors from 1 to 254 may vary.

Since many desktop terminals support 256 colors, it is recommended that custom colors be specified using a terminal's built-in colors: *-setab* and *-setaf.* The default editor values are: *-setab=7* and *-setaf=0.* The utilities *colorhints* and *txcolor,* distributed with the editor, may give hints on selecting values for a terminal.

MACROS

The editor has elementary support for macros. This support is in the form of a "record/play" capability. As documented in the section "COMMANDS", the editor supports "record", "play", and "store"

commands. The command *<cmd>record* begins recording editor commands and typed text, until a second *<cmd>record* command is given. The command *<cmd>-record* will also stop recording; the first version is provided so that it may be assigned to a key via a keyboard description file which can be used both to start and stop a macro recording. The macro that has been recorded may then be replayed by the command *<cmd>play [N]*. The optional count *N* gives the number of times the macro should be replayed; once is the default. The macro thus recorded may be assigned a name by executing *<cmd>store [name]*. The named macros may be replayed by the command *<cmd>$name [N]*. Macros may be saved to a file. This is done via the *update macros* editor command. The file is named, by default, "~/.e_macros". This name may be overridden either by the environment variable EMACROFILE or by means of the "-macrofile=filename" command line option. This file will be read in automatically. The command line option "-nomacros" prevents this. The command *<cmd>undefine macroname* is used to remove a macro definition, which is useful prior to issuing the "<cmd>update macros" command to remove a definition from the *.e_macros* file.

COMMANDS

The following commands are available to the user by opening the command line via the <CMD> key and entering the command. The entered command is parsed and executed when the <RETURN> key is typed.

In the following discussion, pound sign ("#") introduces a comment, brackets ("[","]") indicate options, and vertical bar ("|") indicates optional choices.

There are two types of areas: a range of lines (indicated by <range>) and a marked rectangle of text (indicated by <rectangle>). Some commands will take either type of area, this is indicated by <area>. A <range> can be specified either by marking or by a number followed by "l" for lines or "p" for paragraphs; rectangular areas must be marked. The default range is one line for most commands (fill, adjust and center default to one paragraph).

b | bye | ex | exit [nosave | quit | abort | dump]

Exit the editor. The different exit options affect whether or not the editor updates the edited files or not, and whether the ".e" state files are changed. The following table give the complete exit story.

SavesUpdateRemoveRemove
FilesStateKeys fileChange file

exitXXXX
exit nosave-XXX
exit quit--XX
exit abort---X

exit dump----

blot adjust | close | erase | pick | run | box
-blot adjust | close | erase | pick | run | box
cover adjust | close | erase | pick | run | box
insert adjust | close | erase | pick | run | box
overlay adjust | close | erase | pick | run | box
underlay adjust | close | erase | pick | run | box

All of these commands take the named buffer (one of adjust, close, erase, pick, run, or box) and add or merge it into the current file.

Insert takes the named buffer, moves the existing text to make room for it, and then inserts it at the current text location. It is identical to the -pick command, except that now the other named buffers can also be inserted.

Cover places the named buffer over top of the text without moving anything.

Overlay is like cover, except that only the printing characters in the buffer clobber the original text.

Underlay is the reverse of overlay, i.e. the buffer covers only the non-printing characters of the original text.

Blot is like overlay except that positions in the file corresponding to printing characters in the buffer are erased.

-blot is like blot except that positions in the file corresponding to blanks in the buffer are erased.

Not all of the combinations listed above have been implemented.

box <rectangle>
Draws a box around the rectangle with "+", "|", and "-" characters.

caps <rectangle>
Caps changes the "target text" to all upper case characters (only the characters a-z are affected). The *mark* command is first used to define the "target text".

ccase <rectangle>
Ccase changes all upper case to lower case and vice versa. The affected area is first defined by the *mark* command.

center [<range>] width=n
fill [<range>] width=n
justify [<range>] width=n

These commands all act upon the range of text and then replace them with the modified results. Center centers the text around the column (width/2), fill places as many words as possible on each line, and justify is like fill except that it also provides a smooth right-margin by embedding blanks. The old text is saved in the adjust buffer. As of e19, fill/just no longer filters out CTRL characters, and no longer splits hyphenated words at the end of a line. The command "CMD: set hy" will reinstate splitting.

close [<area>]
-close

Close deletes the text from the file and remembers it in a "close buffer", -close puts the close buffer back into the file at the current cursor position.

command
-command

Usually you issue commands one at a time by typing the <CMD> key, the command, and then <RETURN>. The "command" command places you in a mode where you can keep typing commands without hitting the <CMD> key each time. To return to normal edit mode, issue the -command command.

delete

Tells the editor to delete the current file on exit.

dword
-dword

Delete word. If the cursor is "on" a word, that word is deleted, othwise the next word is deleted. The deleted word can be restored by -dword.

e | edit [<filename>]

If no filename is specified, the alternate file (if any) is brought up in the current window. Otherwise the specified file is displayed. If *filename* begins with "~" or "~/", the appropriate home directory is prepended.

erase [<area>]
-erase

Erase replaces the area by blanks and puts the contents into an "erase" buffer; -erase inserts the contents of this buffer at the current cursor position.

feed [<range>] command-string

A Unix shell is started with the command-string, and the text in <range> is supplied to the shell as the standard input. Any results of the command are inserted before the first line in the range.

goto [b | line-number | e | rb | re]

Goto is a quick way of moving around a file. Goto ’b’ moves to the beginning of the file, ’e’ moves to the end. A line-number goes to that line of the file; ’rb’ and ’re’ move to the beginning or end of a marked range (see the range command). A short-cut to goto line-number *n* is "CMD: *n*<RETURN>".

help

Help will clear the screen and (for some terminal types) show a map of the keyboard layout of the standard keyboard, showing the command tied to each keycap. The user may then type any control character, and a short paragraph documenting the associated command will be shown. Note that the keyboard shown does not reflect any optional keyboard file in use, for example by means of the EKBFILE environment variable.

join | -split
split | -join

Join (or -split) combines two lines, split (or -join) breaks a line in two.

?macros

Lists macros which have been saved by the *store* command.

?macros [X]

Shows the keystrokes which have been stored in macro X by the *store* command.

name <newfilename>

Tells the editor to rename the current file to newfilename when it exits.

open [<area>]

Move existing text to make room for <area> worth of blank lines (or a rectangle of blanks).

pick [<area>]
-pick

Pick copies the <area> to a pick buffer, -pick inserts the pick buffer at the current cursor position.

play [N]

Begin playback of a recording, once or *N* times.

range [<range>]
-range
?range

The range command is another method of limiting the range of some commands (notably the replace, fill, justify and center commands). -range turns it off, ?range tells you the line numbers of the range area.

record
-record

Begin (or end) saving keystrokes for later playback using the *play* command. In recording mode, the only mouse clicks saved in a recording are button clicks while in **-showbuttons** mode. A mouse click in a window area is ignored while recording.

redraw | red

The redraw command is used to erase and redraw the editor windows in case something has happened to it (for example, line noise on dialup lines, or messages from the operator).

replace [<range>] [<option>] /search-string/replace-string/
-replace [<range>] [<option>] /search-string/replace-string/

The replace command searches forward over the range replacing the search-string with the replace-string; -replace searches backwards doing the same thing. The '/' delimiter can be replaced by other symbols if the search or replace strings contain a slash. Any non-control, non-alphanumeric character can be used as the delimiter.

Two options can also be specified, they are *show* and *interactive.* The *show* option allows the user to see the replacements as they occur. The *interactive* option additionally allows the user control over whether or not to make each replacement by displaying the search-string and allowing the user to hit the <REPL> key to do replacement, or the <+/-SRCH> key to skip the replacement.

regexp

-regexp

The "regexp" command enters regular expression mode (for searching and replacing); "-regexp" exits regular expression mode. While in regular expression mode, "RE" is displayed at the bottom of the screen.

run [<range>] command-string

The run command is similar to the feed command, except that the marked text is deleted and replaced by the results of running the command-string.

save <newfilename>

Immediately write a copy of the file to newfilename.

set *option*
?set
used to set various options:
{}on|off, toggles brace matching mode (shortcut)
bracematchon|off, toggles brace matching on/off
brace range *n*sets brace matching range to *n* lines (default: 100)
brace ?shows the brace range value (default: 100 lines)
setting the limit to 0 (or "off") extends the search to the end of file.
brace codingon|off, sets brace matching to "coding" mode where a
search stops at the end of the current function(); assumes that
a curly brace in column one ends a function() definition.
+line *n*sets the <+line> key to *n* lines
-line *n*sets the <-line> key to *n* lines
line *n*sets the <+line> and <-line> keys to *n* lines
+page *n*sets the <+page> key to *n* screens
-page *n*sets the <-page> key to *n* screens
page *n*sets the <-page> and <+page> keys to *n* screens
?displays the values of several options (same as the "?set" command)
bellenable the bell
nofilldota fill cmd with no parameters will stop at a line beginning
with a period (this is the new default)
filldotfill will not stop at lines beginning with a period (as before)
nobelldisable the bell
highlightset mode to: on|off|color|bold|rev
hyenables splitting of hyphenated words by fill/just
nohydisables splitting of hyphenated words by fill/just (default)
left *n*sets the <window left> key to *n* cols
lmar *n*sets the left margin to *n*
mouseon|off enable/disable mouse
right *n*sets the <window right> key to *n* cols
rmar *n*sets the right margin (linewidth) to *n*
window *n*sets the <window left> and <window right> keys to *n* cols
width *n*sets the linewidth to *n* cols (for fill, etc.)
wordelim *mode*sets the action of the <+word> and <-word> keys.
If mode is *whitespace* (the default), words are delimited by blanks and newlines, and the cursor advances to the first character following the delimiter. If mode is *alphanum*, words are delimited by all special characters in addition to blanks and newlines, and the cursor advances to the first alphanumeric character following the delimiter.

The "line", "page", and "window" options may be individually set in different windows.

stop

Use ’job control’ to suspend the editor. Returns control to the shell, resume with the ’fg’ command. Works under most modern operating systems.

store X

Save the current recording in a macro named X for later playback by "<cmd> $X [N]", which inserts the contents of macro X at the current cursor position, once or *N* times. Up to ten macros may be stored for playback.

tab [column ...]
-tab [column ...]

Tab sets tabs in the specified columns; -tab removes tabs in specified columns.

tabs n
-tabs

Tabs sets tabs every nth column, and -tabs removes all tabs.

tabfile tabfilename
-tabfile tabfilename

Tabfile sets tabs (-tabfile clears tabs) at every column listed in the tabfilename.

tag [keyword | + | -]

The tag command is useful only for editing computer source code, not general text. It is used to find the definition of a function or variable name, specified either on the cmd line or whose name is currently under the cursor. It uses a *tags* file generated by the *ctags* utility. This file must be in the current directory in which the editor is run. If no argument is given, the editor will open the file containing the function definition and place the cursor on the name of the function in the definition. If the *tags* file contains multiple matching tags for the target, the command "tag +" will advance to the next definition, and "tag -" will back up to the previous definition. If there is a second window open in the editor, that window will be used to display the function or variable definition, leaving the original window unchanged. This allows the programmer to use the tag facility to look up a definition without changing the current coding position.

Currently, only C language tags are known to work.

track
-track

The track command is used to scroll the main and alternate files together. This is normally used to visually compare the main file and the alternate file by rapidly changing between them and watching what changes on the screen (similar to blink-comparators as used in astronomy). The -track command turns off the tracking mode. As of e19, the TRACK flag has been made part of the state of each window, so that the TRACK mode can be reestablished following an interruption, and the TRACK command has been changed so that it toggles the TRACK mode, in the same way as does INSERT.

undefine macroname

Removes macroname. This command is useful prior to issuing the "update macros" command to remove a macro definition from the *.e_macros* file.

update | -update [inplace | -inplace | -readonly | macros]

The update command tells the editor what to do on exit with the current file. The update command can be used to specify whether or not to break links, the -update command causes any changes to the current file to be ignored. The *-readonly* option is used to enable modifications to files which (1) you own but lack write permission on, or, (2) to override a "-readonly" invocation argument. The *macros* option causes any defined macros to be saved to the macros file as specified in the section MACROS.

w | window [filename]

-w | -window

The window command will create a new window at the current cursor position (as long as the cursor is along the top or left margins). If a filename is specified, that file is displayed in the new window, otherwise the current file is used. The -window command deletes the current window.

wp

-wp

The *wp* command enters word procesing mode, *-wp* exits. A WP is displayed at the bottom of the screen in word processing mode. WP mode enables power typing or "word wrap", where any text that is entered past the right margin (see "set rmar" above) is automatically placed on the following line at the left margin (see "set lmar" above). WP mode also moves the cursor to the left margin in response to a <RETURN>. If WP mode is entered while a marked area is in effect, the boundaries of the marked area become the left and right margin settings. N.B.: Every line break introduced by WP will cause the line involved to be added to the **#o** pseudo-file.

FUNCTION KEYS

E supports a set of *functions* that are normally executed by hitting one of the terminal functions keys, or by typing a specific control- or escape sequence. Details of what *E-functions* are emitted for a specific terminal’s function key layout are hard to document generally, since each site usually tailor’s

this to their liking. Your system administrator should be able to provide a mapping. Also, see the following sections *Standard Editior Keyboard* and *Terminals/Keyboards Supported*. Note that some *E-functions* require first pressing the <CMD> key followed by another <KEY>. Refer to the previous section "COMMANDS" for a discussion of the following functions: <CLOSE>, <OPEN>, <PICK>, <ERASE>, and <MARK> which are normally implemented on function keys. The sequences <CMD><CLOSE>, <CMD><PICK> and <CMD><ERASE> are documented above as *-close, -pick,* and *-erase*.

<LT-ARROW> | <UP-ARROW> | <DN-ARROW> | <RT-ARROW>
: The arrow keys move one space (or line) in the indicated direction. An <UP-ARROW> on the top line of the window (or <DN-ARROW> on the bottom line) scrolls the window down (or up) one line. A <LT-ARROW> at the left edge of a window does nothing. A <RT-ARROW> at the right edge of a window does nothing if option *-stick* is set, otherwise the window is shifted right (16 spaces by default).

<+LINE>, <+LINE>
: scrolls window down/up about 1/4 the window size.

<CMD><+LINE>, <CMD><-LINE>
: makes the current cursor line the top (or bottom) line of the window.

<+PAGE>, <-PAGE>
: scrolls window down/up a full "page".

<CMD><+PAGE>, <CMD><-PAGE>
: moves to the end/beginning of the file.

<+SRCH>, <-SRCH>
: searches in the indicated direction for the next occurrence of the "search string". The *search string* is set by <CMD>search string<+/-SRCH>.

<CMD><+SRCH>, <CMD><-SRCH>
: sets the "search string" to the current cursor word, and performs a search in the indicated direction.

<+TAB>, <-TAB>
: tabs in indicated direction.

<+WORD>, <-WORD>
: moves to next/previous word.

<ALT> switch to alternate file, also "<CMD>e<RETURN>".

<BS> backspace, moves left and erases previous character. <CMD><BS> deletes text from the beginning of the line up to the current cursor character. In *insert mode*, the rest of the line is moved left.

<CHWIN> change to next window.

<CTRLCHAR>
used to enter non-ascii characters (\000-\037) in text.

<DELCHAR>
deletes cursor character.

<CMD><DELCHAR>
deletes text from the cursor to the end of the line.

<HOME> moves to upper left corner of window.

<INSERT> toggles *insert* mode.

<INT> Interrupts the operation of various functions: e.g., searches.

<MARK> used to define a range of lines or rectangular area of text. A simple <MARK> will start by defining one *line*. Moving the cursor up/down will extend the defined range of *lines* (as will other keys like <+LINE> <+PAGE>). When full lines are *marked*, the message "MARK *n*" is displayed at the bottom of the screen. Rectangular areas of text are marked by first marking one or more lines, then moving the cursor left or right. When a rectangular area is marked, the message "MARK *lines*x*cols*" is displayed.

<CMD><MARK>
cancels any mark in affect.

<REPL> Executes the "replacement" while doing an interactive replace command.

<RETURN>
Positions cursor at column 1 on the next line. A <RETURN> on the bottom line of a window is equivalent to a <+LINE>. In WP mode, a <RETURN> positions the cursor at the current *left margin* on the next line, which may automatically shift the window left.

<WINLEFT>, <WINRIGHT>
Shifts the window left/right (16 spaces by default).

<CMD><WINLEFT>
Shifts the window all the way back to column 1.

<CMD><WINRIGHT>
Shifts the window as far right as necessary to display the end of the line.

STANDARD EDITOR KEYBOARD

There is now a "standard" E keyboard, that is designed to be usable on all video display terminals. Either say "e -keyboard=standard" or "setenv EKBD standard" before running E to select this keyboard. This is also the keyboard that E will use if there is no specific knowledge of your type of terminal compiled into the editor.

This keyboard layout is designed to be used on terminals with no function keys. "^H" means control-H, and "^X-^U" means control-X followed by control-U. The ALT entry gives you a choice of "^/" or "^_" because one or the other or both will not work on some terminals. ("_" is correct ASCII.)

\!.3rmk "\$1"
\!.nr 3crow 0
\!.3rvpt "\$1"
??E STANDARD KEYBOARD??

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 1
\!.3rvpt "\$1"

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 2
\!.3rvpt "\$1"
(extend)

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 3
\!.3rvpt "\$1"
+LINE

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 4
\!.3rvpt "\$1"
+PAGE

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 5

```
\!.3rvpt "\$1"
+SRCH

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 6
\!.3rvpt "\$1"
+TAB

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 7
\!.3rvpt "\$1"
+WORD

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 8
\!.3rvpt "\$1"
-LINE

\!.3rvpt "\$1"
```

```
\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 9
\!.3rvpt "\$1"
-PAGE

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 10
\!.3rvpt "\$1"
-SRCH

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 11
\!.3rvpt "\$1"
-TAB

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"
```

```
\!.3rmk "\$1"
\!.nr 3crow 12
\!.3rvpt "\$1"
-WORD

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 13
\!.3rvpt "\$1"
ALT

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 14
\!.3rvpt "\$1"
BSP

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 15
\!.3rvpt "\$1"
CHG WIN
```

```
\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 16
\!.3rvpt "\$1"
CLOSE

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 17
\!.3rvpt "\$1"
CMD

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 18
\!.3rvpt "\$1"
CTRLCHAR

\!.3rvpt "\$1"

\!.nr 3brule 1
```

```
\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 19
\!.3rvpt "\$1"
DELCHAR

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 20
\!.3rvpt "\$1"
DOWN

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 21
\!.3rvpt "\$1"
ERASE

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 22
```

\!.3rvpt "\$1"
HOME

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 23
\!.3rvpt "\$1"
INSERT

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 24
\!.3rvpt "\$1"
INT

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 25
\!.3rvpt "\$1"
JOIN

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 26
\!.3rvpt "\$1"
LEFT

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 27
\!.3rvpt "\$1"
MARK

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 28
\!.3rvpt "\$1"
OPEN

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

```
\!.3rmk "\$1"
\!.nr 3crow 29
\!.3rvpt "\$1"
PICK

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 30
\!.3rvpt "\$1"
REPL

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 31
\!.3rvpt "\$1"
RETURN

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 32
\!.3rvpt "\$1"
RIGHT
```

```
\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 33
\!.3rvpt "\$1"
SPLIT

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 34
\!.3rvpt "\$1"
TABS

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 35
\!.3rvpt "\$1"
UP

\!.3rvpt "\$1"

\!.nr 3brule 1
```

```
\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 36
\!.3rvpt "\$1"
WIN LEFT

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 37
\!.3rvpt "\$1"
WIN RIGHT

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 38
\!.3rvpt "\$1"


\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 39
```

```
\!.3rvpt "\$1"


\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 40
\!.3rvpt "\$1"


\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 41
\!.3rvpt "\$1"


\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 42
\!.3rvpt "\$1"


\!.3rvpt "\$1"
```

```
\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 43
\!.3rvpt "\$1"


\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 44
\!.3rvpt "\$1"


\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 45
\!.3rvpt "\$1"


\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"
```

\!.3rvpt "\$1"

TERMINALS/KEYBOARDS SUPPORTED

This is installation specific information with the actual terminal types defined varying across the host cpu's versions of the editor. Now that the editor can be driven from termcap and the standard keyboard, many sites will wish to delete specific terminal type support to save memory. (N.b: These days, the amount of memory thus saved is several orders of magnitude less than anybody would conceivably be worried about). Code is provided with E for the following terminals and may be compiled into your version.

Also, note that the EKBFILE environment variable may be used to redefine the keyboard at will on an individual user basis.

\!.3rmk "\$1"
\!.nr 3crow 0
\!.3rvpt "\$1"
??DESCRIPTION ??

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 1
\!.3rvpt "\$1"

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 2

```
\!.3rvpt "\$1"
Ann Arbor S001901/2

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 3
\!.3rvpt "\$1"
New ann arbor (Model Q2878)

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 4
\!.3rvpt "\$1"
Ann Arbor Ambassador

\!.3rvpt "\$1"

\!.nr 3brule 1


\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 5
\!.3rvpt "\$1"
 and Ann Arbor XLs

\!.3rvpt "\$1"
```

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 6
\!.3rvpt "\$1"

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 7
\!.3rvpt "\$1"
Beehive MicroBee

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 8
\!.3rvpt "\$1"
Termcap terminal

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 9
\!.3rvpt "\$1"
INTERACTIVE Systems Intext2

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 10
\!.3rvpt "\$1"
Lear Siegler ADM3a

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 11
\!.3rvpt "\$1"
Liberty Freedom 100

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 12
\!.3rvpt "\$1"
Heathkit H19 & H89

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 13
\!.3rvpt "\$1"
INTERACTIVE Systems Intext

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 14
\!.3rvpt "\$1"
Perkin Elmer 1251

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 15
\!.3rvpt "\$1"
Standard keyboard

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 16
\!.3rvpt "\$1"
DEC VT100 (needs termcap)

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 17
\!.3rvpt "\$1"
Lear Siegler ADM31

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 18
\!.3rvpt "\$1"
Lear Siegler ADM42

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 19

\!.3rvpt "\$1"
Concept 100

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 20
\!.3rvpt "\$1"
Datamedia DM4000

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rmk "\$1"
\!.nr 3crow 21
\!.3rvpt "\$1"
Wyse 50

\!.3rvpt "\$1"

\!.nr 3brule 1

\!.3rvpt "\$1"

\!.3rvpt "\$1"

PLATFORMS SUPPORTED

This version of the editor has been ported to, and tested on, Linux systems (specifically CentOS) and

Mac systems (specifically Mojave and Catalina). It is not a "clickable" app; it must be run in a terminal window.

The Linux version has been run under the Linux subsystem (specifically Ubuntu) on Windows 10, but has not been extensively tested there. Or, really, tested at all.

To build the distribution on a given platform, read the README.md file in the GitHub distribution. In brief, look in the top-level Makefile file to find the target for your platform ("sys5" for Linux, "bsd" for Mac). Then, read the e19/Makefile file to see what reconfiguration must be done to build your target. "e" was built long, long before the *config* utility was developed.

STARTUP FILE

As a means of individually tailoring various options on startup, E will run commands specified in a "profile" file. The order in which E decides which "profile" file to run, is: command line option -profile=*file*, environment variable EPROFILE20, then environment variable EPROFILE, a "safe" *./.e_profile*, *~/.e_profile20*, and finally, *~/.e_profile*. ("*./.e_profile*" is deemed to be safe if you are not the super-user, you own the file, and it's not writable by everyone else.) The EPROFILE20 environment variable and *~/.e_profile20* file are searched first in case the original RAND editor branch is also in use. The original editor does not support the new features of the current version. These new features, if present in a profile file, would cause the old version of the editor to exit with an error. Therefore, for convenience, these new features may be placed in a profile file that the old version of the editor will not search for. If the first line of the profile file begins with the keyword "options:", then the rest of the line is treated as if typed as initial arguments to E. The options must currently all be on one line.

Profile format notes:

- blank lines and lines beginning with # are ignored
- E function keys are denoted by "<keyword>"; use "\<" to insert a "<" in text; "<#keyword>" may be used to repeat *keyword* #-times
- keywords:

<+line> <caps> <int> <right>*
<+page> <ccase> <join> <split>
<+sch> <cchar> <left>* <srtab>
<+tab> <chwin> <mark> <tab>
<+word> <close> <mouse> <up>*
<-line> <cmd> <open> <wleft>
<-mark> <dchar> <pick> <wp>
<-page> <down>* <play> <wright>
<-sch> <dword> <record> <null>
<-tab> <edit> <redraw> <undef>

<-word> <erase> <regex>
<bksp> <home> <replace>
<brace> <insmd> <ret>
(*=cursor motion)

- essentially anything that can be typed at E can be inserted in the ".e_profile". Thus, application specific editor scripts can be created.
- note that various CMD options end with a <ret>, others don't; e.g., "<cmd>tabs 4<ret>", but "<cmd><+page>".
- E looks for a "profile" file and examines the "options:" line *before* processing any command line arguments, thus command line options override any options specified in the "options:" line. *Caveat*: if the -profile=file argument is used, any "options:" in a profile file will have already been processed. This is a "cart before the horse" problem....
- any format errors will terminate reading of the profile file, and continue the E session;
- example:

```
options: -regex -stick
<cmd>tabs 4<ret>
<cmd>set window 40<ret>
```

There is an "examples/" directory at the top level of the distribution which contains sample *.e_profile* and *.e_profile20* files.

BUGS

By far the worst problem with E is its treatment of tabs. E is a "whitespace" editor, all whitespace is considered equal. On any line that E modifies, it will compress multiple initial blanks to tabs, strip trailing blanks, and convert any embedded tabs to blanks. This is acceptable for most editing, but some programs expect real tab characters as input, for example tbl(1) and some nroff commands. However, the new options **-lit, -nostrip, -tabs**, and **-blanks** alleviate most of these problems.

The "Terminal" application on the Mac has problems with use of the *-bgcolor* and *-fgcolor* arguments. The application does its own processing, and prevents the specified colors from being used. The *-setab* and *-setaf* arguments work correctly. An alternative is to use another terminal application on the Mac. *iTerm2* is known to work.

Resizing terminal windows in the parent operating system is not supported.

SEE ALSO

N-2056-1: SELF-TEACHING GUIDE TO RAND'S TEXT PROCESSOR, 1986, The Rand Corporation

N-2239-1: THE RAND EDITOR, e, Version 19, 1986, The Rand Corporation.

These are available from *https://rand.org*.

Two other documents worthy of note are R-2000-ARPA, *A Guide to NED: A New On-Line Computer Editor, 1977,* and R-2176-ARPA,

SPONSOR

The Rand Corporation.