NAME

aee - another (easy) editor

SYNOPSIS

aee [options ...] [file[s]]
rae [options ...] file[s]
xae [options ...] [file[s]]
rxae [options ...] file[s]

DESCRIPTION

aee and *xae* are non-modal editors, that is, the user does not need to switch from one mode to another to insert and delete text. The editor is always in text mode. Control sequences and function keys are used to perform the editing functions. In the case of *xae*, the mouse can also be used to position the cursor, and perform text selection and pasting.

rae and *rxae* are the same as *aee* and *xae* respectively, except that they are restricted to editing the file(s) named on the invoking command line. No other files may be opened or written, nor may shell operations be performed.

The arrow keys (up, down, left, right) may be used to move the cursor. If the keyboard is so equipped, the **prev** and **next** keys will move the cursor to the previous and next pages, respectively.

The **gold** key is a key that is used to provide alternate behavior for a key, much like a gold function key on a calculator keyboard. So, for example, function key number 7 (**f7**) may be used for initiating a search, while pressing the **gold** key and then **f7** will produce a prompt for the string to be searched for. The gold function can be assigned to any assignable key. By default, the keys **f1** and **control-g** (**^g**) are assigned the gold function.

The rest of the keys by default will behave as described below.

OPTIONS

The following are accepted as options when starting *aee* or *xae*:

-e	Turn off echo from initialization file.
-i	Turn off info window.
-j	Turn off journaling.
-n	Turn off highlighting of menu and info window borders.
-r	Recover from failed edit session (using journal file from that session).
-t	Expand tabs to spaces.
+*number*	Move cursor to start of line *number*.

The following additional options are available for *xae* only:

-fn *font*	Specify a font for use with *xae*. Resource name **BaseFont**.
-fg *color*	Specify a foreground color (color of text). Resource name **ForegroundColor**.
-bg *color*	Specify a background color. Resource name **BackgroundColor**.
-geometry =*columns***x***lines*	Specify a geometry for *xae*. Resource name **Geometry**.

The *xae* specific options may be specified in the **X-Windows Version 11** defaults file for the user. See your X-Windows documentation for more details.

Function Keys

aee uses the terminal's function keys to perform editing tasks. By default, only keys **f1** through **f8** are defined. (Although a terminal may have function keys, the keys may not send unique sequences that can be recognized by software, or may require particular settings in order to be useful. If in doubt, ask your system administrator.)

center; l l l. KEY NAME NORMAL FUNCTION GOLD FUNCTION

F1 GOLD GOLD F2 undel character undel line F3 delete word undel word F4 advance word begin of line F5 search search prompt F6 mark copy F7 cut paste F8 advance line command

Control Keys

Control keys are the regular alphabetic keys pressed in conjunction with the **control** key (sometimes spelled **CTRL**). To press control-a (also noted as **^a**), first press **control**, then press '**a**', so that both keys are pressed at the same time.

center; l l l. KEY NAME NORMAL FUNCTION GOLD FUNCTION

Control A ascii code match Control B bottom of text append Control C command clear to eol control D begin of line prefix Control E command Control F forward format Control G GOLD GOLD Control H backspace Control J carriage-return Control K delete character undelete char Control L delete line undelete line Control M carriage-return Control N next page next buffer Control O end of line Control P prev page prev buffer Control R redraw screen reverse Control T top of text Control U cut paste Control V mark copy Control W delete word undelete word Control X search search prompt Control Y advance word prev word Control Z replace replace prompt Control [(Escape) menu

Menu Operations

Pop-up menus can be obtained by pressing the **escape** key (or **^[** if no **escape** key is present). When in the menu, the escape key can be used to leave the menu without performing any operations. Use the up and down arrow keys, **^u** and **^d** or the **space** or **backspace** keys to move to the desired items in the menu, then press **return** to perform the indicated task.

The main menu in *aee* is as follows:

leave editor
If changes have been made, the user will get a menu prompting whether or not the changes should be saved.

help Displays a help screen, with all of the keyboard operations and commands.

edit Pops up a menu to allow the user to **mark**, **copy** marked text, **cut** marked text, or **paste** previously marked text.

file operations
Pops up a menu for selecting whether to read a file, write to a file, or save the current contents of the editor, send the contents of the editor to a print command (see the section **Initializing aee from a file**), as well as recovering from a previous edit session (see **Recovery** below).

redraw screen
Provides a means to repaint the screen if the screen has been corrupted.

settings
Shows the current values of the operating modes, and margins. By pressing return when the cursor is on a particular item, the value can be changed. To leave this menu, press the **escape** key. (See **Modes** below.)

search/replace
Pops up a menu in which the user may choose to enter a string to search for, or search for a string already entered, or perform a replace string operation.

miscellaneous

Pops up a menu that allows the user to format the current paragraph, execute a shell command, or check the spelling of the text in the current buffer.

Commands

Some operations require more input than one key can provide, or do not need to be so easily accessed. These commands are entered using the prompt provided by pressing the **command** (ˆE or ˆC or gold f8) key. The line commands are:

autoformat *	Turn automatic paragraph formatting on.
noautoformat *	Turn automatic paragraph formatting off (default).
buffer [*name*]*	This command will move from the current buffer to the buffer with the name given by the user. If the buffer did not previously exist, it is created. If no parameter is given, then the name of the current buffer is displayed. Note: a buffer created with this command is not associated with a file, nor is it journalled.
case*‡	Specifies that the case of each letter is to be taken into account in the search operation.
nocase*‡	Specifies that there is no distinction between upper and lower case during search and replace operations (default).
cd *directory*	Change directory
character	Displays the ascii code of the character the cursor is on.
define [gold] *key string**	Assigns all of the string following the key definition to the key specified by the user. The commands allowed in the string are described in the SYMBOLS section.
delete	Deletes the current buffer (the initial buffer may not be deleted).
echo *string* †	Echoes the string to the terminal during startup of *aee*.
edit *filename*	Edit another file. A new buffer will be created in which the named file can be edited. If no file is specified, a temporary buffer name is created for use within *aee*. If journalling is on for the initial file being edited, journalling will occur for the new edit buffer as well.
eight*	If your terminal has an eight bit character set, then use this command to cause *aee* to send the eight bit value to your terminal. The default condition is to display the eight bit character as the decimal value of the character between angle brackets.
noeight*	If your terminal does not have an eight bit character set, use this command to cause *aee* to display eight bit characters as the decimal value of the character between angle brackets (251 is displayed as <251>). This is the default condition.
exit[!]‡	Exit the current edit session writing out the main buffer to the file name used in entering the editor. The optional exclamation mark allows you to leave without editing the rest of the files in the list of files specified when the edit session was invoked.
expand*‡	Causes spaces to be inserted when the tab key is pressed. Spaces fill to the next tab stop.
noexpand*‡	Tabs are *not* replaced with spaces (default).
file	Display the name of the file being edited.

help‡	Provides the user with information about using the editor while in the editor.
helpfile†	The location and name of the file containing help information.
indent*‡	When creating a new line by pressing the carriage return, the new line will have the same indentation (number of spaces and tabs) as the previous line.
noindent*‡	Turns off the indent mode (default).
info *	Turn info window on (default).
noinfo *	Turn info window off.
journaldir	Specifies the path to the directory where journal files are to be created.
justify	Justify the right side of the text when using the **format** function to format a paragraph.
nojustify	Turn off right justification of a paragraph (default).
line	Displays the current line number.
literal*‡	Causes characters in search string to be matched one-to-one with characters in the text.
noliteral*‡	Allows metacharacters in the search string (default).
margins *‡	Causes left and right margins to be observed (set using **leftmargin** and **rightmargin**).
nomargins *‡	Allows lines to be any length (disregards the margin settings).
leftmargin [*number*] *	Set the left margin to *number*. If no number is specified, then the current value is displayed.
rightmargin [*number*] *	Set the right margin to *number* (**nowrap** must be set for margin setting to be observed). If no number is specified, then the current value is displayed.
overstrike*‡	Causes characters to overstrike or replace existing characters instead of inserting.
nooverstrike*‡	Causes characters to be inserted into line at current cursor position without replacing existing characters (default).
print	Sends the contents of the current buffer to the printer. The command that is used can be specified in the *init.ae* file, see section **Initializing aee From A File**.
printcommand †	Allows the setting of the print command (default: "lp").
pwd	Display the current directory.
quit[!]‡	Quit the current edit session without writing a file. The optional exclamation mark has the same meaning as for the **exit** command.
read *file*	Read a file into the current buffer after the cursor.
resequence	Renumber the lines.
save	Save the contents of the main buffer to the file being edited.
show [gold] *key*	Displays the function(s) assigned to the specified key.
status*‡	A status line is displayed on the bottom line of the screen.
nostatus*‡"	Turns off the status line (default).
stops *number* *	Tabs will be spaced every *number* spaces, unless other tabs are set using the **tabs** command.

tabs [*stops* ...] *	Sets tabs to *stops*. After the last user defined tab stop, tabs are the normal sequence of every eight columns, or as set using the **stops** command. The first column is 0.
untabs *stops* ... *	Removes the specified tab stops.
windows*‡	This command specifies whether or not buffers are displayed on the screen simultaneously. If **windows** (default) is specified, then buffers exist on the screen together.
nowindows*‡	This command specifies that there is only one buffer on the screen at a time.
write *file*	Write the current buffer out to the specified file.
0123456789	Enter a number to go to the line corresponding to that number.
+ or **-** *number* ‡	Moves forward or back the number of lines specified.
[<*inbuff*] [>*outbuff*] **!**command	Execute the command following the exclamation mark in the UNIX shell. The shell used is the one specified in the shell variable **SHELL** in the user's environment, or **/bin/sh** if **SHELL** is not defined. You may send data from the buffer *outbuff* (or the current buffer if *outbuff* is not specified) out to the shell by using the right angle bracket (>). You may read into *inbuff* (or the current buffer if *inbuff* is not specified) by using the left angle bracket (<) as shown. The data read in from the command will be placed after the current cursor location in the buffer

* may be used in init file, see section **Initializing aee From A File**

† only used in initialization file

‡ may also be assigned to a key using the **define** command

Search and Replace

aee's search facility provides several abilities. The user may choose for the search to be case sensitive, or ignore the case (upper or lower) of a character (**nocase** is the default). The user may also choose **literal**, or **noliteral** (the default) modes for the search facility. The **literal** mode interprets the search string literally, **noliteral** means that some characters (called metacharacters) have special meaning, as described below:

center; l l. symbol meaning

ˆ beginning of line $ end of line \\x interpret 'x' literally [abc] T{ match a single character in the text to one in brackets T} [a-z] T{ match a single character in the text to one in range a-z T} [ˆabc] T{ match a single character in the text that is not within the brackets after 'ˆ' ('ˆ' means 'not') T} * T{ match any sequence of characters, useful in middle of string with known beginning and end, but variable middle T}

The carat (ˆ) within the square brackets ([]) means that the search will match any characters *not* within the brackets. The carat *must* be the first character after the opening bracket.

The asterisk (*) may be useful when searching for a string to which you know the beginning and end, but not what characters (if any) or how many may be in the middle. The first character after the asterisk should not be a metacharacter (a character with special meaning).

The replace facility uses the same modes as the search facility. The prompt for the replace operation shows the syntax for the input:

/string1/string2/

where the slash ('/') may be replaced by any character that is not in the search or replacement string, and "string1" is to be replaced by "string2". When in **noliteral** mode, the search string may be placed in the replacement string by using the ampersand ('&'), like so:

/old/abc&123/

Where "old" will be inserted between "abc" and "123".

Recovery

If for some reason an edit session is interrupted, it is possible to recover the work done in the session. This is accomplished via the information stored in the journal file, which is a record of the changes made to the text in the buffer while in the editor. To recover a session in which a file named *foo* was being edited, use the command:

aee -r *foo*

This is only possible if the **-j** option was not used, since the **-j** option turns journaling off.

It is also possible to start *aee* with no arguments, and then to browse the journal files. This is accomplished through the menus. To perform this task, bring up the menu by pressing the **Esc** key, select *file operations*, then select *recover from journal*. You should then be presented with a list of files to recover.

Key Definitions

The function keys and control sequences (alphabetic keys pressed with the control key) may be defined by the user to perform any of the functions described below.

The user may assign more than one function to each key, as long as each one is separated by one or more spaces. The following describes the functions of the keys and how the user may redefine the keyboard during the edit session on the command line. The same syntax is used in the *initialization file*.

Note that the '^' is typed by the user in the following examples, and is **not** generated by pressing the control key and letter, and that f2 is entered by typing an 'f' and then a '2'.

Examples:

define ^b dl

will define the key control b to have the function delete line.

define gold ^b udl

assigns the function undelete line to GOLD control b.

define f2 /this is an inserted string/ cr

will cause the string between the delimiters (/) to be inserted followed by a carriage-return whenever the function key f2 is pressed.

If you wish to have a key that deletes to the end of line without appending the next line to the end, you may make the following key definition:

define f3 dl cr left

This set of functions will delete to the end of line and append the next to the end of the line, then insert a line at the cursor, and then move the cursor back to the end of the previous line, the position where you started. This may of course be assigned to any valid key.

Symbols

The following symbols, as well as the commands noted by (‡) in the list of commands may be assigned to keys using the **define** command.

l l. Symbol Description _ menu pop up menu dl delete line dc delete character dw delete word und T{ undelete last thing deleted, keeps last 128 things deleted T} udl undelete line udc undelete character udw undelete word eol end of line bol begin of line bot begin of text eot end of text np next page pp previous page nb next buffer pb previous buffer gold gold il insert line psrch search prompt srch search prp replace prompt rp replace fwd forward (search forward of cursor) rev reverse (search before cursor) al advance line aw advance word pw previous word format format paragraph mark mark text prefix T{ mark text and place *before* existing text in paste buffer T} append T{ mark text and place *after* existing text in paste buffer T} cut cut marked text copy copy marked text pst paste previously cut or copied text unmark T{ unmark text, doesn't affect previous paste buffer contents T} ac ascii character mc match (), {}, [], or <> cmd command up up arrow down down arrow left left arrow right right arrow rd redraw screen bck backspace cr carriage return /,. T{ the first non-alpha character will act as a separator to allow for single line text insertion, the second occurrence of the same character will end the insertion T}

Initializing aee From A File

aee checks for a file named *init.ae* in */usr/local/lib*, *.init.ae* in the user's home directory, then for *.init.ae* in the current directory. If the file exists, it is read and initializes *aee* to the parameters as defined in the file. By having initialization files in multiple places, the user may specify settings for global use, and then supplement these with customization for the local directory. The parameters allowed in the *init.ae* file are key definitions, turning off windowing, case sensitivity, literal searching, eight bit characters, as well as the ability to echo strings to the terminal (see the **Commands** section for the commands allowed in the initialization file). An example follows:

```
define ^z rp
define gold ^z prp
define f3 und
define f4 unmark
define k0 srch
define gold k0 psrch
case
printcommand lp -dlaser
echo \033&jB
```

The above example assigns the command **replace** to control-z, and **replace prompt** to gold control-z, as well as setting *aee* to be sensitive to the case of characters during search and replacement operations. It also defines the function keys **f3** and **f4** to be **undelete** and **unmark** respectively. The **print** command will send its output to the device 'laser' through the UNIX command **lp**. A string is **echo**ed to the terminal which will "turn on" the user function keys on an HP terminal.

The user may wish to **echo** strings to the terminal when starting *aee* **to set up the terminal or other devices, so the echo** facility is provided. **Echo** is applicable *only* in the initialization file. No quotes are required around the string to be echoed. Characters may be literal or escaped (using the backslash convention). The **-e** option on the invoking command line turns off the **echo** operation. This may be useful if you normally use one type of terminal (and echo strings for its use), but occasionally use another terminal and do not wish the strings to be **echo**ed.

Operations allowed in the initialization file are noted in the list of commands with an asterisk (*).

Shell Escapes

Sometimes it is desirable to execute shell commands outside of the editor. This may be accomplished by pressing a key assigned to the **command** function (**^C**, **^E**, or **gold F8**), and then entering an exclamation mark (!) followed by the shell command(s) to be executed. It is possible to send data from the editor to be processed by a shell command and/or read data from a shell command into a buffer in the editor. The format for this is as follows:

<*inbuff* >*outbuff* !*command*

where *inbuff* is the name of the buffer to receive the data and *outbuff* is the name of the buffer to output to the shell command. By omitting the name of the buffer, the current buffer will be used. For example, if you have a list of names and wish them sorted, you could use the UNIX command *sort*. If you wished to view them while in the current edit session, you could use the following sequence:

<sorted >list !sort

where *list* is the name of the buffer containing the unsorted list, *sorted* is the name of the buffer to contain the sorted list, and *sort* is the name of the UNIX command to be executed. The data read in from the command will be placed after the current cursor location in the receiving buffer. If the specified buffer does not exist when the command is entered, it will be created. You should be sure of the spelling of the name of the buffer to be the input of the command if you are specifying one.

PRINT Command

The **print** command allows you to send the contents of the current buffer to a command specified by using the **printcommand** operation in the initialization file. The default is '**lp**', using the default device.

If you choose to specify something other than the default command, the command should be able to take its input from *stdin*, since *aee* will set up a pipe to feed the information to the command.

Paragraph Formatting

Paragraphs are defined for *aee* by a block of text bounded by:

- Begin or end of file.
- Line with no characters, or only spaces and/or tabs.
- Line starting with a period ('.') or right angle bracket ('>').

A paragraph may be formatted two ways: explicitly by choosing the **format paragraph** menu item, or by setting *aee* to automatically format paragraphs. The automatic mode may be set via a menu, or via the initialization file.

There are three states for text operation in *aee*: free-form, wrap, and automatic formatting.

"Free-form" is best used for things like programming. There are no restrictions on the length of lines, and no formatting takes place. Margins are not enabled for this state.

"Wrap" allows the user to type in text without having to worry about going beyond the right margin (the right and left margins may be set in the **settings** menu, the default is for the right margin to be the right edge of the terminal). This is the mode that allows the **format paragraph** menu item to work. The "observe margins" entry in the "settings" menu allows the user to toggle this state, as well as the **margin** and **nomargin** commands (see **Commands** above).

"Automatic formatting" provides word-processor-like behavior. The user may type in text, while *aee* will make sure the entire paragraph fits within the margins every time the user inserts a space after typing or deleting text. Margins must also be enabled in order for automatic formatting to occur. The "auto paragraph format" item in the "settings" menu allows the user to toggle this state, as well as the commands **autoformat** and **noautoformat**.

Modes

Although *aee* is a 'modeless' editor (it is in text insertion mode all the time), there are modes in some of the things it does. These include:

tabs to spaces
: Tabs may be inserted as a single tab character, or replaced with spaces.

case sensitive search
: The search operation can be sensitive to whether characters are upper- or lower-case, or ignore case completely.

literal search
: Allows the user to specify whether regular expressions are to be used for searching or not.

observe margins
: The left and right margins can be observed, or not.

info window
: A window showing the keyboard operations that can be performed can be displayed or not.

status line
: Display the file name, position in the file, and selected status indicators.

auto indent
: The editor can be set to automatically indent the newly inserted line the same as the previous line, or not (primarily useful for programming).

overstrike
: Toggle text insertion or overstrike modes.

auto paragraph formatting
: While typing in text, the editor can try to keep it looking reasonably well within the width of the screen.

multi windows
: Allow multiple buffers to be displayed at the same time, or only a single buffer at a time.

You may set these modes via the initialization file (see above), with a menu (see **Menu** above), or via commands (see **Commands** above).

Mark, Cut, Copy, and Paste

To move large chunks of text around, use the control key commands **mark** (^V or f6), **cut** (^U or f7), and **paste** (gold ^U or gold f7). These commands allow you to mark the text you wish to use so that it may be put in the paste buffer. The paste buffer differs from the buffers mentioned below in that you may not move to it, and that it is only used for these operations. Once the text has been placed in the paste buffer, you may move your cursor wherever you wish and insert the text there, as many times you wish, anywhere you want.

Simply move the cursor to the start of a section of text you wish to mark, and press the key assigned the control key function **mark** (^V or f6). Move the cursor over the text you wish to place in the paste buffer. The text between the cursor position at which you pressed the **mark** key and the current position will be highlighted. Once you have marked all of the text you wish to place in the paste buffer, press the key for **cut** (^U or f7) or **copy** (gold ^V or gold f6). The **cut** operation deletes the text from the buffer in which the text is contained, and the **copy** operation simply places the text in the paste buffer without deleting it. Now you may move to another section and use the **paste** function to insert it as many times as you wish.

If you wish to copy several sections of text that are not adjacent to each other, you may use the **append** function (gold ^B) to put the text you will mark at the end of the current paste buffer contents, or the **prefix** function (gold ^D) to place the newly marked text before the current paste buffer contents when you **copy** or **cut**. The **prefix** and **append** functions are used in place of the **mark** function.

Sometimes you may start marking text, then decide to cancel the operation. It is possible to do that by using the **unmark** operation. The **unmark** operation is not assigned to any key by default.

Buffer Operations

aee allows you to examine more than one file in the editor during one edit session. This mechanism is known as buffers, and the first buffer is called **main**. Buffers may or may not be viewed simultaneously on the screen depending upon the wishes of the user. The default condition allows for the buffers to co-exist on the screen, but this may be changed by using the command **nowindows**. To return to the default condition, enter the command **windows**, and buffers will co-exist on the screen. You may switch between having windows and not having windows at any time without losing information in the buffers. *aee* will remind you that you have buffers if you attempt to leave the editor without deleting them.

Buffers are created when you enter the command **buffer** with a single argument, which is used as the buffer's name. This command not only creates the new buffer, but it moves the cursor to the new buffer. This is also the way to move to a buffer which already exists. Buffer movement is also facilitated by the control key commands **nb** (gold ^N) and **pb** (gold ^P), short for **next buffer** and **previous buffer**. This is especially useful if you mistype or forget the name of any of the buffers you have created and have windowing turned off.

The command **buffer** without any arguments displays the name of the current buffer.

Buffers can be deleted by using the command **delete**. Simply move to the buffer to be deleted and use the command **delete**. You **cannot** delete the first buffer, called **main**. Note that all commands operate in the buffer in which your cursor is operating, so you may use the **write** and **read** commands to save or read a file into the current buffer.

WARNINGS

A journal file will not be created if the user does not have write permission to the directory in which the file is to reside.

If the file to edit does not exist, and is to be created in a directory to which the user does not have write permission, *aee* will exit with an error without editing the file.

The automatic paragraph formatting operation may be too slow for slower systems.

Writing the information to the journal during some operations may be rather disk I/O intensive, which may impact performance noticably on slower systems. Journal files may also grow considerably during an edit session, which may be a concern if there is little disk space available. Turning off journaling may be useful if performance is slow, but turning off journaling will mean the loss of data should a system or network failure occur while using the editor.

CAVEATS

THIS MATERIAL IS PROVIDED "AS IS". THERE ARE NO WARRANTIES OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Neither Hewlett-Packard nor Hugh Mahon shall be liable for errors contained herein, nor for incidental or consequential damages in connection with the furnishing, performance or use of this material. Neither Hewlett-Packard nor Hugh Mahon assumes any responsibility for the use or reliability of this software or documentation. This software and documentation is totally UNSUPPORTED. There is no support contract available. Hewlett-Packard has done NO Quality Assurance on ANY of the program or documentation. You may find the quality of the materials inferior to supported materials.

Always make a copy of files that cannot be easily reproduced before editing. Save files early, and save often.

FILES

/usr/local/lib/init.ae
$HOME/.aeeinfo
$HOME/.init.ae
.init.ae

AUTHOR

The software *aee* and *xae* was developed by Hugh Mahon.