1 THE TEDIT TEXT EDITOR

TEdit is a window-based, modeless text editor, capable of handling fonts and some rudimentary formatting. Text is selected with the mouse, and all editor operations act on the current selection.

There are two ways to start TEdit: With an explicit call to the function TEDIT, or from the background menu. At top level, you can call

```
(TEDIT TEXT WINDO W )
```

where TEXT is the thing you want to edit, and WINDO W is an optional argument specifying the window you want to do the editing in. To start a fresh editing window, specify a TEXT of NIL. Otherwise, TEXT may be the name of an existing le, a string to be edited, or an arbitrary [MKSTRING-able] Lisp object.

The text is displayed in an editing window, and may be edited there (see below). There will be a one-line-high prompting window across the top of the editing window; it is used to ask for le names, search strings, and the like.

The TEdit option in the background menu opens an empty editing window; you may either type in the text you want, or use the Put menu option (below) to bring in a le.

1.1 Selecting Text

TEdit works by operating on ‘‘selected’’ pieces of text. Selected text is highlighted in some way, and may have a caret ashing at one end. Insertions go where the caret is; deletion and other operations are applied to the currently selected text.

Text is selected using the mouse. There are two regions within an edit window: The area containing text, and a ‘‘line bar’’ just inside the left edge of the window. While the mouse is inside the text region, the cursor is the normal up-and-left pointing arrow. When the cursor moves into the line bar, it changes to an up-and-right pointing arrow. Which region the mouse is in determines what kind of selection happens:

The LEFT mouse button always selects the smallest things. In the text region, it selects the character you’re pointing at; in the line bar, it selects the single line you’re pointing at.

The MIDDLE mouse button selects larger things. In the text region, it selects the word the cursor is over, and in the line bar it selects the paragraph the cursor is next to.

The RIGHT button always extends a selection. The current selection is extended to include the character/word/line/paragraph you are now pointing at. For example, if the existing selection was a whole-word selection, the extended selection will also consist of whole words.

There are special ways of selecting text which carry an implicit command with them:

If you hold the CTRL key down while selecting text, the text will be shown white-on-black. When you release the CTRL key, the selected text will be deleted. You can abort a CTRL-selection: Hold down a mouse button, and release the CTRL key. Then release the mouse button.

Holding the SHIFT key down while making a selection causes it to be a ‘‘copy-source’’ selection. A copy source is marked with a dashed underline. Whatever is selected as a copy source when the SHIFT key

is released will be copied to where the caret is. This even works to copy text from one edit window to another. You can abort a copy: Hold down a mouse button, and release the SHIFT key. Then release the mouse button.

Holding the CTRL and SHIFT keys down while making a selection causes it to be a ''move'' selection, which is marked by making it veverse video. Whatever is selected as a ''move'' source when the CTRL and SHIFT keys are released will be moved to where the caret is. This even works to move text from one edit window to another. You can abort a move: Hold down a mouse button, and release the CTRL and SHIFT keys. Then release the mouse button. If the variable TEDIT.EXTEND.PENDING.DELETE is non-NIL, extending a selection will display the selection as white-on-black. The next time something is typed (or if text is copied or moved there), the inverted text will be deleted rst. This provides an easy way of replacing text.

1.2 Editing Operations

Inserting text: Except for command characters, whatever is typed on the keyboard gets inserted where the caret is. The BS key and control-A both act as a backspace, deleting the character just before the caret. Control-W is the backspace-word command.

Deleting Text: Hitting the DEL key causes the currently-selected text to be deleted. Alternatively, you can use the CTRL-selection method described above.

Copying Text: Use SHIFT-selection, as described above.

Moving Text: Use CTRL-SHIFT-selection.

Undoing an edit operation: The top blank key is the Undo key (the KEYBOARD key on the 1108). It will undo the most recent edit command. Undo is itself undo-able, so you can never back up more than a single command.

Redoing an edit operation: The ESC key is the Redo key (the forward-arrow key on the 1108). It will redo the most recent edit command on the current selection. For example, if you insert some text, then select elsewhere, hitting ESC will insert a copy of the text in the new place also. If the last command was a delete, Redo will delete the currently-selected text; if it was a font change, the same change will be applied to the current selection.

The command menu: You can get command menus by moving into the edit window's title region and hitting the RIGHT or MIDDLE mouse buttons. RIGHT gets the usual menu of window commands. MIDDLE gets a menu of editor commands:

Put — Causes an updated version of the le to be written. Tedit will ask you for a le name, o ering the existing name (if any) as the default. When the le name is o ered, you may type ^E to abort the operation.

Get — Lets you read in a new le to edit, *without saving the one you were working on.* You'll be asked for a le name in the prompt window. Instead of a le name, you may type ^E to abort the operation.

Include — Lets you copy the contents of a le into the edit window, inserting it where the caret is. Tedit will ask you for a le name in its prompt window. Instead of a le name, you may type ^E to abort the operation.

`Quit` Causes the editor to stop without updating the file you're editing. If you haven't saved your changes, you'll be asked to confirm this.

`Find` Asks for a search string, then hunts from the caret toward the end of document for a match. Selects the first match found; if there is none, nothing happens. The search is case-sensitive; i.e. ''Foo'' will not be found with the search string ''FOO''.

`Substitute` Asks for a search string and a replacement string. Within the current selection, all instances of the search string are replaced by the replacement string. If you wish, TEdit will ask you to confirm each replacement before actually doing it.

`Looks` Changes the character looks of the selected characters: The font, character size, and face (bold, italic, etc.). Three menus will pop up in sequence: One to select the font name, one to select the face, and one to select the size. You may select an option in each menu. If, for example, you want to leave the character size alone, just click the mouse outside the size menu. In general, any aspect of the character looks that you don't change will remain the same.

`Hardcopy` Prints the document to your default press or InterPress printer, with 1 inch margins all around. The variable `DEFAULTPRINTINGHOST` controls which kind of printer TEdit will send to.

`Press File` Creates a Press or InterPress file of the document, with 1 inch margins all around. The file format is also controlled by `DEFAULTPRINTINGHOST`.

`Expanded Menu` Opens a large menu that has three parts: A TEdit operations section, a character-looks menu, and a paragraph looks (*i.e.,* formatting) menu. This expanded menu has fill-in blanks for some fields (like what the `Find` command should hunt for), and has on-off-neutral buttons to control character properties like boldness. This is described below.

1.3 TEdit Tedit Expanded Menu

Selecting the item ''Expanded menu'' from TEdit's title-bar menu creates a large free-form menu on top of your editing window. The expanded menu contains selectable menu buttons and places for you to type text (e.g., what to search for when you do a FIND). The menu is itself a TEdit window, so the usual editing operations work with one change. Some parts of the menu can't be selected or operated on; they're protected. The places you can select are: menu buttons, the margin ruler (see below), and between pairs of curly braces, so: {}.

Menu buttons appear in bold; every menu button which needs to ask for text has a pair of {} associated with it, e.g., the line

`Quit` `Hardcopy` server: `{}` copies: `{}`

has two buttons on it. The ''`Quit`'' button needs no further arguments, while the ''`Hardcopy`'' button can take two arguments: the name of the server to print to and the number of copies to print. Fill the text in before hitting the menu button.

The TEdit operation menu looks like this:

TEdit Tedit Expanded Menu

```
Quit   CloseMenu   All
Get   {}   Put   {}   Include   {}
Find   {}   Substitute   {}   for   {}
Hardcopy  server: {}  copies:  {}
```

The Get, Put, Include, and Find buttons all require a text argument, which must be typed in before you hit the corresponding menu button. Substitute requires two arguments, the second being the search string, and the rst being the replacement. The Hardcopy button takes two optional text arguments. If you specify a server name, the hardcopy will be sent there. If you leave the brackets empty, TEdit uses DEFAULTPRINTINGHOST as usual. You may also specify how many copies of the document you want; if you don't put anything in the ''copies'' eld, you get one copy.

The Quit, CloseMenu and All buttons need no additional arguments. Quit stops the current editing session; All causes the entire document to be selected. This is useful for making global substitutions or changes to character looks. CloseMenu closes the expanded menu, if you want to get rid of it without stopping the edit entirely.

The Character Looks Menu looks like this:

```
Character Looks Menu:      APPLY    SHOW
Props:    Bold  Italic  Underline  StrikeThru  Overbar
TimesRoman  Helvetica  Gacha  Cream  Other
Size:  {}  Normal  Superscript  Subscript  distance:  {}
```

Generally speaking, you select the text you want to change, set the entries in this menu up as you want the text to appear, then make the change by hitting the APPLY button.

If you have a piece of text whose looks you want to copy, select the text and hit the SHOW button. The menu will be lled in to match that text's looks. You can then APPLY it elsewhere, perhaps after modifying things slightly.

The second line of the menu is a list of character properties which can be modi ed independently. Each of the menu buttons has three states: If the button appears white-on-black, that property will be turned on; If the button appears with a diagonal line, that property will be turned o ; If the button appears black-on-white, that property will be left alone.

Why is it useful to leave a property alone? Suppose you have a paragraph in Times Roman with some bold and some italic in it. If you want to change the font to Helvetica without changing the boldness or italicness, you can do so. The third line of the menu is a list of font-family names. You can select among them: selecting one family deselects any others. You can also select no family by mouse buttoning between two of the families. If you APPLY with no font family selected, the text will be left in whatever font family it was.

The last line of the menu lets you set the font's size, and specify any superscripting or subscripting. Fill in the ''Size:'' eld with a number, and APPLYing will change all the selected characters to that size. Leave it empty, and the characters will retain their existing sizes.

For character o sets, you have three choices: Normal characters lie on the baseline; Superscript characters lie above the baseline by the distance you specify (2 points by default); Subscript characters lie below the baseline by the distance you specify (2 points by default). As with font family names, you may mouse in the space between options to neutralize the choice. APPLYing with a neutral choice leaves characters with the super- and subscripting they had, if any.

The Paragraph Looks Menu looks like this:

```
Paragraph Looks Menu:      APPLY     SHOW
Left  Right  Centered  Justified
Line leading:  {}  Paragraph leading:  {}
Tab Type:  Left Right Centered Decimal    Default Tab Size:  {}
```

Below this menu, if you scroll it up far enough, is a solid black rectangle, used for setting indentations, and a ruler, used for setting tab stops.

As with the Character Looks Menu, you select the text you want to change, set the entries in this menu up as you want the text to appear, then make the change by hitting the APPLY button.

If you have a paragraph whose looks you want to copy, select the text and hit the SHOW button. The menu will be lled in to match that text's looks. You can then APPLY it elsewhere, perhaps after modifying things slightly.

The second line of the menu is for specifying how the paragraph margins are to be justi ed. A Left justi ed paragraph has a ragged right margin, but is justi ed ush with the left margin. A Right justi ed paragraph has a ragged left margin, but is justi ed ush on the right. A Centered paragraph is centered between the two margins. A Justi ed paragraph is set ush with both the left and right margins.

The space between lines in a paragraph is called ''line leading''. You can specify it, in units of printer's points. You can also leave space in front of a paragraph (without using extra carriage returns) by specifying ''paragraph leading,'' also in units of printer's points.

You set paragraph margins using the margin ruler on the bottom. There are three margin values: The left margin for the paragraph's rst line, the left margin for the rest of the paragraph, and the paragraph's right margin. The margin ruler is has three sensitive areas, one for each margin value. Margins are measured in printer's picas (6 to the inch), with a grain of 1/2 pica. There are 12 points to the pica. Plans exist for allowing di erent units (and granularity) in the ruler.

The rst- line left margin is controlled by the top half of the ruler, left end. To move it, push a mouse button near the left edge, and hold it. Moving the mouse pulls the margin along with it; the margin ruler always shows the current values of the margins. If you push the RIGHT mouse button over the margin, it becomes neutral; i.e., APPLYing the paragraph menu won't change the rst- line left margins of any paragraphs. The rest-of-paragraph left margin is controlled by the bottom half of the ruler, left end. You move it (and neutralize it) the same way.

Likewise for the right margin, which is controlled by the right end of the margin ruler. There are a couple of di erences here. First, you can set the right margin to 0, which will create a '' oating'' right margin (one that follows the right edge of the edit window or of the printed page). This is signalled by a margin ruler that is as wide as the window, but shows a value of 0 at its right end.

Since the editing window may be narrower than the document, you can also set the right margin beyond the edge of the window, by pulling it with the mouse, and pulling past the window edge. A right margin you can't see is represented by a double wavy line at the right edge.

You can also set tab stops using the margin ruler. The space below the ruler markings is sensitive to all three mouse buttons, and is used to represent tab stops.

To set a tab, you rst nned to choose what kind of tab you want, using the line starting with ''Tab Type:.'' Make your choice of tab type the same way you'd choose a font family. Left tabs are regular

typewriter type tabs; Right tabs take the succeeding text and push it so it is ush- right against the tab stop location; Centered tabs cause the succeeding text to be centered about the tab stop; Decimal tabs (not implemented) cause the succeeding text to have its decimal point lined up on the tab stop. Tab stops are shown in the margin ruler as small arrows with suggestive tails.

To create a new tab stop, use the middle mouse button. In the region below the ruler markings (and the numbers!), point to where you want the tab to be, and press the middle mouse button. The tab should appear; as long as you hold the button down, the tab will follow the mouse around, so you can adjust its location. To move a tab stop, point at it and press the left mouse button. As long as you hold it down, the tab stop will follow the mouse. To delete a tab stop, point at it and press the right mouse button.

1.4 TEdit Functional Interface

The top-level entry to TEdit is:

`(TEDIT TEXT WINDOW DONTSPAWN PROPS)` [Function]

`TEXT` may be a (litatom) file name, an open `STREAM`, a string, or an arbitrary [`MKSTRING`-able] Lisp object. The text is displayed in an editing window, and may be edited there. If `TEXT` is other than a file name, a `STREAM`, or a string, `TEDIT` will call `MKSTRING` on it, and let you edit the result.

If `WINDOW` is `NIL`, you will be prompted to create a window. If `WINDOW` is non-`NIL`, `TEDIT` will use it as the window to edit in. If `WINDOW` has a title, `TEDIT` will preserve it; otherwise, `TEDIT` will provide a descriptive title for the window.

`TEDIT` will normally spawn a new process to run the edit, so you can edit in parallel with other work; indeed, it is possible to have several editing windows active on the screen. You can have the editing done in your process and have TEdit return the result of the edit by calling `TEDIT` with `DONTSPAWN` set to `T`.

`PROPS` is a prop-list-like collection of properties which control the editing session. The following options are possible:

`FONT` The default character looks (font, size, etc.) to be used in the edit window. This can be a `FONTDESCRIPTOR`, or a property list of character looks properties such as `TEDIT.LOOKS` would accept, or a `CHARLOOKS` data structure describing the character looks.

`QUITFN` A function (or list of functions) to call when the user `Quit`s. If any of the functions is `T` or returns `T`, the user will not be asked to confirm the `Quit` even if he'll potentially lose something. If any of the functions returns `DON'T`, the `Quit` is aborted before the user is asked for confirmation.

`LOOPFN` A function to be called each time thru the character-read loop.

`CHARFN` A function to be called for each character typed in.

`SELFN` A function to be called each time a mouse selection is made in this edit window.

TERMTABLE If you want characters displayed other than TEdit's default way, set this to a Terminal table with the correct settings.

READTABLE If you want command characters which are local to this edit session, set this to a Read table with the appropriate settings.

BOUNDTABLE If you want word breaks to happen other than the default way, set this to a Read table with the appropriate settings.

READONLY If the value of this property is non-NIL, then the edit window will be read-only, i.e., you can only shift-select in it.

CACHE If the value of this property is non-NIL, then the file being edited will be cached locally instead of being read as needed from the remote server.

SEL Tells what text should be selected initially. This can be a SELECTION (see below) describing the selected text, or a character number, or a two-element list of first character number and number of characters to select.

MENU Describes the menu to be displayed when the MIDDLE mouse button is pressed in the edit window's title region. If it is a MENU, that menu will appear. If it is a list of menu items, a new menu will be constructed.

AFTERQUITFN
A function to be called *after* TEdit has quit. This can be used for cleanup of side-effects by TEdit client programs.

TITLEMENUFN
A function to get called instead of bringing up the usual TEdit command menu when the user LEFT- or MIDDLE-buttons in the edit window's title region.

PARALOOKS The default paragraph looks to be used for paragraphs in this document. This can be either a FMTSPEC data structure, or a property list of paragraph formatting information such as TEDIT.PARALOOKS would accept.

CARETLOOKSFN
A function that is called whenever new caret looks are being set.

LEAVETTY If this is non-NIL, TEdit will not take control of the keyboard when it is started. Instead, it will wait until you first button in the editing window with the mouse.

PROMPTWINDOW
A window that is to be used for unscheduled user interactions, in place of the prompting window that TEdit usually provides. If this is the atom DON'T, no window will be provided, and the main prompt window will be used.

Any PROPS specified will be appended to the front of whatever is the value of TEDIT.DEFAULT.PROPS; respecified properties will override anything in the defaults. This provides client applications with a way to set default edit properties.

1.5 TEdit Functional Interface

The ''Text Stream'' Data Structure

TEdit keeps a STREAM which describes the current state of the text you're editing. You can use most of the usual stream operations on that stream: BIN, SETFILEPTR, GETFILEPTR, and GETEOFPTR do the usual things. BOUT inserts a character in the stream just in front of the next character you'd read if you BINned. You can get the stream by calling (TEXTSTREAM EditWindow).

If you need to save the state of an edit, you can save this stream. Calling TEDIT with the stream as the TEXT argument will let you continue from where you left off.

There is a datatype called TEXTSTREAM which defines several fields that are of interest within the stream:

TEXTOBJ — The TEXTOBJ which describes the edit session.

PIECE — The PIECE which describes the text at the file pointer.

The ''Text Object'' Data Structure

TEdit keeps a variety of other information about each edit window, in a data structure called a TEXTOBJ. Field TEXTOBJ of a text STREAM points to the associated TEXTOBJ, which contains these fields of interest:

\WINDOW — The edit window which contains the text. If this is NIL, there is no edit window for this text.

SEL — The most recent selection made in this text.

SCRATCHSEL — A scratch SELECTION, used by the mouse handler for the edit window, but otherwise available for scratch use.

TEXTLEN — The current length of the edited text.

STREAMHINT — Points to the text STREAM which describes the text.

EDITFINISHEDFLG — If this is non-NIL, TEdit will halt after the next time through the keyboard polling loop. No check will be made for unsaved changes. Unless it it T, the value of EDITFINISHEDFLG will be returned as the result of TEdit.

The ''Selection'' Data Structure

The selected text is described by an object of type SELECTION, whose fields are as follows:

CH# — The character number of the first character in the selection. The first character in the text being edited is numbered 1.

CHLIM — The character number of the last character in the selection. Must be CH#.

DCH — The number of characters in the selection. If DCH is zero, then no characters are selected, and the Selection can be used only to describe a place to insert text.

ONFLG — Tells whether the Selection is indicated in the edit window. If T, it is; if NIL, it's not.

\TEXTOBJ — The TEXTOBJ that describes the selected text. You can use this to get to the Stream itself.

X0 — The X position (edit-window-relative) of the left edge of the first selected character.

Y0 — The Y position of the bottom of the first selected character (not the character's base line, the bottom of its descent).

XLIM — The X position of the right edge of the last character selected. If DCH is zero (a ''point'' selection), XLIM= X0.

YLIM — The bottom of the last character in the selection.

DX — The width of the selection. If DCH is zero, this will be also.

SELOBJ — This is for a future object-oriented editing interface.

POINT — Tells which side of the selection the caret should appear on. It will be one of the atoms LEFT and RIGHT.

SET — T if this selection is currently valid, NIL if it is obsolete or has never been set.

SELKIND — What kind of selection this is. One of the atoms CHAR, WORD, LINE, or PARA.

HOW — A TEXTURE, which will be used to highlight the selecton.

HOWHEIGHT — How high the highlighting is to extend. A selection's highlight starts at the bottom of the lowest descender, and extends upward for HOWHEIGHT pixels. To always get highlighting a full line tall, set this to 16384.

HASCARET — T if this selection should have a caret flashing next to it, NIL otherwise.

1.6 TEdit Interface Functions

TEdit exports the following functions for use in custom interfaces:

(OPENTEXTSTREAM TEXT WINDOW START END PROPS) [Function]

Creates a text STREAM describing TEXT, and returns it. If WINDOW is specified, the text will be displayed there, and any changes to the text will be reflected there as they happen. You will also be able to scroll the window and select things there as usual. TEXT may be an existing TEXTOBJ or text STREAM. If START and END are given, then only the section of TEXT delimited is edited. PROPS is the same as for TEDIT.

Given the STREAM, you can use a number of functions to change the text in an edit window, under program control. The edit window gets updated as the text is changed.

(TEDIT.SETSEL STREAM CHqorSEL LEN POINT PENDINGDEL? LEAVECARETL OOKS) [Function]

Sets the selection in STREAM . If CHqorSEL is a SELECTION, it is used as-is. Otherwise, CHqorSEL is the rst character in the selection, and LEN is the number of characters to select (zero is allowed, and gives just an insertion point). POINT tells which side of the selection the caret should come on. It must be one of the atoms LEFT or RIGHT.

If PENDINGDEL? is non-NIL, the selection will be a pending-delete selection the selected text will be deleted at the next type-in (or if text is copied or moved there). Otherwise, the selection will be a normal selection.

Normally, the act of making a selection sets the ''caret'' looks the looks for any characters typed at the caret. This can be suppressed by passing in a non-NIL LEAVECARETL OOKS .

(TEDIT.GETSEL STREAM) [Function]

Returns the SELECTION which describes the current selection in the edit window described by STREAM .

(TEDIT.SHOWSEL STREAM ONFL G SEL) [Function]

Lets you turn the highlighting of the selection SEL on and o. If ONFL G is T, the selection SEL in STREAM will be highlit in the edit window; if NIL, any highlighting will be turned o. If SEL is NIL, it defaults to the current selection in STREAM .

(TEDIT.SEL.AS.STRING STREAM SEL) [Function]

Returns the currently-selected text as a string. If SEL is non-NIL, the text it describes will be returned.

(COERCETEXTOBJ STREAM TYPE) [Function]

Converts a text stream, TEXTOBJ, or edit window into another form, speci ed by TYPE . If TYPE is the atom STRINGP, it will return a string (with any formatting and font information stripped out). If TYPE is FILE, it will create a le on the CORE device which contains the text including all formatting and font information.

(TEDIT.INSERT STREAM TEXT CHqorSEL LOOKS DONTSCR OLL) [Function]

Inserts the string TEXT into STREAM , as though it had been typed in. CHqorSEL tells where to insert the text: If it's NIL, the text goes in where the caret is. If it's a FIXP, the text is inserted in front of the corresponding character (The rst character in the stream is numbered 1). If it's a SELECTION, the text is inserted accordingly.

If the LOOKS argument is provided, it must be a font descriptor. The inserted text will appear in that font.

Normally, TEdit scrolls the editing window so that each change is visible as it is made. If you want the window left where it is instead, the DONTSCR OLL argument should be non-NIL.

(TEDIT.DELETE STREAM CHqorSEL LEN) [Function]

Deletes text from STREAM. If CHqorSEL is a SELECTION, the text it describes will be deleted; if CHqorSEL is a FIXP, it is the character number of the first character to delete. In that case, LEN must also be present; it is the number of characters to be deleted.

(TEDIT.INCLUDE TEXTOBJ FILE START END) [Function]

Performs the TEdit ''Include'' command, inserting the text from file FILE into TEXTOBJ. If START and END are supplied, only the specified portion of the file is included.

(TEDIT.FIND STREAM TEXT STARTq ENDq) [Function]

Searches for the next occurence of TEXT inside STREAM. If STARTq is present, the search starts there; otherwise, the search starts from the caret. If ENDq is present, the search will end at that character; otherwise, it ends at the end of the text. If a match is found, TEDIT.FIND returns the character number of the first character in the matching text. If no match is found, it returns NIL.

(TEDIT.HARDCOPY STREAM FILE DONTSEND BREAKPAGETITLE SERVER) [Function]

Sends the text contained in STREAM to the printer. If a file name is given in FILE, the press file will be left there for you to use. If DONTSEND is non-NIL, the file will not be sent to the printer; use this if you only want to create a press file for later use.

If BREAKPAGETITLE is non-NIL, it is used as the title on the ''break page'' printed before the text.

You can specify the print server where the hardcopy is to be sent, using the SERVER argument; if it is NIL, TEdit uses DEFAULTPRINTINGHOST.

(TEDIT.LOOKS STREAM NEWLOOKS SELORCHq LEN) [Function]

Changes the character looks of selected characters, e.g., the font, character size, etc. SELORCHq can be a SELECTION, an integer, or NIL. If SELORCHq is a SELECTION, the text it describes will be changed; if it is a FIXP, it is the character number of the first character to changed. In that case, LEN must also be present; it is the number of characters to be changed. A SELORCHq of NIL will use the current selection.

NEWLOOKS is a property-list-like description of the changes to be made. The property names tell what to change, and the property values describe the change. Any property which isn't changed explicitly retains its old value. Thus, it is possible to make a piece of text all bold without changing the fonts the text is in. The possible list entries are as follows:

FAMILY — The name of the font family. All the selected text is changed to be in that font.

FACE — The face for the new font. This may be in either of the two forms acceptable to FONTCREATE: a list such as (BOLD ITALIC REGULAR), or an atom such as MRR.

WEIGHT — The new weight for the font. This must be one of LIGHT, MEDIUM,

or BOLD. Specifying this *disables* the FACE parameter.

SLOPE — The new slope for the font. This must be one of REGULAR or ITALIC. Specifying this *disables* the FACE parameter.

EXPANSION — The new weight for the font. This must be one of CONDENSED, REGULAR, or EXPANDED. Specifying this *disables* the FACE parameter.

SIZE — The new point size.

UNDERLINE — The value for this property must be one of the atoms ON or OFF. The text will be underscored or not, accordingly.

OVERLINE — The value for this property must be one of the atoms ON or OFF. The text will be overscored or not, accordingly.

STRIKEOUT — The value for this property must be one of the atoms ON or OFF. The text will be struck through with a single line or not, accordingly.

SUPERSCRIPT — A distance, in points. The text will be raised above the normal baseline by that amount. This is mutually exclusive with SUBSCRIPT.

SUBSCRIPT — A distance, in points. The text will be raised above the normal baseline by that amount. This is mutually exclusive with SUPERSCRIPT.

PROTECTED — The value for this property must be one of the atoms ON or OFF. If it is ON, the text will be protected from mouse selection and from deletion.

SELECTPOINT — The value for this property must be one of the atoms ON or OFF. If a character has this property, the user can make a point selection just after it, even if the character is also PROTECTED.

INVISIBLE — The value for this property must be one of the atoms ON or OFF. If a character has this property, the character will not appear on the screen or on hardcopy.

(TEDIT.PARALOOKS STREAM NEWLOOKS SELORCHq LEN) [Function]

Changes the paragraph looks of selected paragraphs, e.g., the margins, line leading, etc. SELORCHq can be a SELECTION, an integer, or NIL. If SELORCHq is a SELECTION, the text it describes will be changed; if it is a FIXP, it is the character number of the first character to changed. In that case, LEN must also be present; it is the number of characters to be changed. A SELORCHq of NIL will use the current selection. In all cases, TEDIT.PARALOOKS operates on *whole paragraphs*. If any portion of a paragraph is included in the selection, the entire paragraph's looks will be changed.

NEWLOOKS is a property-list-like description of the changes to be made. The property names tell what to change, and the property values describe the change.

Any property which isn't changed explicitly retains its old value. Thus, it is possible to make a paragraph indented without changing its tab stops. The possible list entries are as follows:

QUAD — One of LEFT (for ush- left, ragged-right), CENTERED (for centered lines), RIGHT (for ush- right, ragged-left), or JUSTIFIED (for ush- left and -right).

1STLEFTMARGIN — The left margin for the rst line of the paragraph, in points.

LEFTMARGIN — The left margin for the rest of the paragraph, in points.

RIGHTMARGIN — The right margin for all lines of the paragraph, in points. If this value is 0, one gets a ‘‘ oating’’ right margin, which adjusts to the width of the edit window or paper.

TABS — This is a CONS pair, whose CAR is a relative tab width and whose CDR is a list of absolute tab stops. A tab advances the cursor to the next absolute tab stop to the right of the current position. Should there be no absolute tab stop to the right of the cursor, the cursor is advanced by the relative tab width. The relative tab width defaults to .5 inches (= 36 pts). Each absolute tab stop is a CONS pair with the car being the position, and the CDR being one of LEFT, RIGHT or CENTER. This value indicates how the word following the tab will be justi ed with respect to the tab. For instance, LEFT indicates that the left edge of the word following the tab will be at the tab position indicated in the CAR. For a RIGHT tab, the right edge of the word following the tab would have been located at the position indicated in the CAR. CENTER indicates that the word following the tab will be centered at the position in the CAR.

LINELEADING — The space to be left before each line of the paragraph, in points.

PARALEADING — Additional space to be left before the rst line of the paragraph, in points.

POSTPARALEADING — Additional space to be left after the last line of the paragraph, in points.

(TEDIT.QUIT STREAM VALUE) [Function]

STREAM must be the text stream associated with a running TEdit. TEDIT.QUIT causes the editing session to end. If VALUE is given, it is returned as TEdit's result; otherwise, TEdit will return the usual result. The user is not asked to con rm his desire to stop editing.

(TEDIT.KILL STREAM) [Function]

STREAM must be the text stream, TEXTOBJ, or edit window associated with

a running TEdit. TEDIT.KILL kills the TEdit process, and cleans up its data structures. It does not cause TEdit to return a result.

(TEDIT.ADD.MENUITEM MENU ITEM) [Function]
Adds a menu ITEM to MENU. This will update the menu's image so that the newly-added item will appear the next time the menu pops up. This is only guaranteed to work right with pop-up menus which aren't visible.

(TEDIT.REMOVE.MENUITEM MENU ITEM) [Function]
Removes a menu ITEM from MENU. This will update the menu's image so that the newly-added item will appear the next time the menu pops up. This is only guaranteed to work right with pop-up menus which aren't visible. ITEM may be either the whole menu item, or just the indicator which appears in the menu's image.

(TEXTOBJ STREAM/WINDO W) [Function]
Given a text stream, or a TEdit editing window, returns the associated TEXTOBJ.

(TEXTSTREAM TEXTOBJ/WINDO W) [Function]
Given a TEXTOBJ or a TEdit editing window, returns the associated text stream.

(TEDIT.FORMATTEDFILEP STREAM) [Function]
Tells whether a given text stream is plain text (result is NIL) or must be stored as a special TEdit-format file (result is one of the atoms CHARLOOKS, PARALOOKS, or IMAGEOBJ, depending on the amount of formatting information that must be stored).

(TEDIT.CARETLOOKS STREAM FONT) [Function]
The looks of newly-typed characters are controlled by the looks that are ''attached to the caret''. This function lets you set those looks for a given document. FONT is either a font descriptor or a CHARLOOKS. Any text inserted or typed in thereafter will appear in that font (or with those looks).

(TEDIT.STREAMCHANGEDP STREAM RESET?) [Function]
Returns T if the text represented by the STREAM has been modified since it was last saved. If RESET? is non-NIL, then the change indicator will be reset i.e., TEdit will then believe that the text is unchanged, and will not ask for confirmation of the Quit and Get operations.

(TEDIT.NORMALIZECARET STREAM SEL) [Function]
Makes sure that the caret is visible in the editing window; if not, the document is scrolled to place the caret on the top line of the window. This is normally controlled by the existing selection for the given text stream. However, if SEL is specified, it is used to decide the caret's location.

(COPYTEXTSTREAM STREAM CROSSCOPY) [Function]
Makes a fresh copy of the text stream STREAM. If CROSSCOPY is non-NIL, the new stream will not share structure with the old one it can be edited without affecting the original stream.

(TEDIT.SELECTED.PIECES TEXTOBJ SEL CROSSCOPY PIECEMAPFN FNARG1 FNARG2) [Function]
Returns a list of PIECEs that describe the text selected in the selection SEL out of

the document TEXTOBJ . If CR OSSCOPY is non-NIL, the pieces will be copied. PIECEMAPFN , if given, is applied to each piece in turn, and the value it returns is used in place of the piece (or its copy).

(TEDIT.PROMPTPRINT TEXTOBJ MSG CLEAR?) [Function]
Prints a message in the TEdit prompting window associated with the given TEXTOBJ . If CLEAR? is non-NIL, the window will be cleared rst.

1.7 User-function ‘‘Hooks’’ in TEdit

TEdit provides a number of hooks where a user-supplied function can be called. To supply a function, attach it to the edit window under the appropriate indicator, using WINDOWPROP. Every user-supplied function is APPLYed to the text STREAM which describes the text. Some of these functions can also be supplied using the PR OPS argument to TEDIT or OPENTEXTSTREAM; the descriptions below contain the details.

TEDIT.QUITFN [Window Property]
A function to be called whenever the user ends an editing session. This may do anything; if it returns the atom DON'T, TEdit will not terminate. Any other result permits TEdit to do its normal cleanup and termination. This can also be supplied using the PROPS argument to TEDIT or OPENTEXTSTREAM.

TEDIT.AFTERQUITFN [Window Property]
A function to be called after the user ends an editing session. This may perform any cleanup of side e ects that you desire. This can also be supplied using the PROPS argument to TEDIT or OPENTEXTSTREAM.

TEDIT.CMD.LOOPFN [Window Property]
A function that gets called, for e ect only, each time through TEdit's main command loop. This can also be supplied using the PROPS argument to TEDIT or OPENTEXTSTREAM.

TEDIT.CMD.CHARFN [Window Property]
A function that gets called, for e ect only, once for each character typed into TEdit. The character code is passed to the function as its second argument. This can also be supplied using the PROPS argument to TEDIT or OPENTEXTSTREAM.

TEDIT.CMD.SELFN [Window Property]
A function that gets called, each time the user tries to select something with the mouse: (SELFN TEXTOBJ SELECTION SELECTMODE FINAL?). It is called once for each tentative selection (e.g., while the mouse button is still down, but gets moved), and once for e ect only for the nal selection. The new SELECTION is passed as the function's second argument, and an atom describing the kind of selection (one of NORMAL, COPY, MOVE, PENDINGDEL (for an extended selection that will be deleted on type-in), or DELETE) as the third. When the function is being called with a candidate selection, FINAL? will be the atom TENTATIVE; when being called with the nal selection, FINAL? is the atom FINAL.

When the function is called with a candidate selection, it may veto that selection by returning the atom DON'T. This can be used to limit selections to items of interest.

If a selection is vetoed, the old selection will remain highlighted; the effect is that of the user being unable to move the selection from its old location.

This can also be supplied using the PROPS argument to TEDIT or OPENTEXTSTREAM.

TEDIT.PRESCROLLFN [Window Property]
Called just before TEdit scrolls the edit window.

TEDIT.POSTSCROLLFN [Window Property]
Called just after TEdit scrolls the edit window.

TEDIT.OVERFLOWFN [Window Property]
Called when TEdit is about to move some text off-screen. This function may handle the text overflow itself (say by reshaping the window), or it may let TEdit take its normal course. If the function handles the problem, it must return a non-NIL result. If TEdit is to handle the overflow, the value returned must be NIL.

TEDIT.TITLEMENUFN [Window Property]
Called whenever the user presses the LEFT or MIDDLE mouse button in the edit window's title region. Can also be supplied using the PROPS argument to TEDIT or OPENTEXTSTREAM. Normally, this is the function TEDIT.DEFAULT.MENUFN, which brings up the usual TEdit command menu.

CARETLOOKSFN [Window Property]
Called whenever TEdit is about to set the caret looks for an edit window. This function, called as (CARETLOOKSFN TEXTOBJ NEWLOOKS) may perform whatever checking it likes, and then return either the atom DON'T, meaning that the caret looks are not to be changed, NIL, meaning that NEWLOOKS should be used as the caret looks, or a new CHARLOOKS which will be used as the caret looks.

Note: if this function returns a new CHARLOOKS, it must *not* be a smashed version of NEWLOOKS.

TEdit also saves pointers to its data structures on each edit window. They are available for any user function's use.

TEXTOBJ [Window Property]
The TEXTOBJ which describes the current editing session.

TEXTSTREAM [Window Property]
The text STREAM which describes the text of the document.

1.8 Changing the TEdit Command Menu

You may replace the MIDDLE-button command menu with one of your own. When you press the MIDDLE button inside an edit window's title region, TEDIT calls the value of the TEDIT.TITLMENUFN window property with the window as its argument. Normally, what gets called is TEDIT.DEFAULT.MENUFN, but you may change it to anything you like.

TEDIT.DEFAULT.MENUFN brings up a menu of commands. If the edit window has a property TEDIT.MENU, that menu is used. If not, TEdit looks for the window property TEDIT.MENU.COMMANDS

(a list of menu items) and constructs a menu from that. Failing that, it uses TEDIT.DEFAULT.MENU.

This means that you can control the command menu by setting the appropriate window properties. Alternatively, you may add your own menu buttons to the default menu, TEDIT.DEFAULT.MENU.

```
(TEDIT.ADD.MENUITEM TEDIT.DEFAULT.MENU ITEM)
```

will add ITEM to the TEdit menu. Menu items should be in the form (NAME (QUOTE FUNCTION)), where NAME is what appears in the menu, and FUNCTION will be applied to the text stream, and can perform any operation you desire.

Finally, you may *remove* menu items from the default menu, by doing

```
(TEDIT.REMOVE.MENUITEM TEDIT.DEFAULT.MENU ITEM)
```

ITEM can be either a complete menu item, or just the text that appears in the menu; either will do the job.

1.9 Variables Which Control TEdit

There are a number of global variables which control TEdit, or which contain state information for editing sessions in progress:

TEDIT.EXTEND.PENDING.DELETE [Variable]
: If this is non-NIL, extending a selection makes it into a pending-delete selection. See the selection section.

TEDIT.DEFAULT.FONT [Variable]
: A FONTDESCRIPTOR. This is the font for displaying TEdit documents which don't specify their own font information.

TEDIT.DEFAULT.FMTSPEC [Variable]
: A paragraph-looks description. This contains the default looks for a paragraph.

TEDIT.SELECTION [Variable]
: A SELECTION. This is the most recent regular selection made in *any* TEdit window.

TEDIT.SHIFTEDSELECTION [Variable]
: A SELECTION. This is the most recent SHIFT-selection made in *any* TEdit window.

TEDIT.MOVESELECTION [Variable]
: A SELECTION. This is the most recent CTRL-SHIFT-selection made in *any* TEdit window.

TEDIT.READTABLE [Variable]
: A read table, this is used to translate typed-in characters into TEdit commands. See the section on TEdit readtables. This can be overridden using the READTABLE property argument to TEDIT.

TEDIT.WORDBOUND.READTABLE [Variable]
: The read table which controls TEdit's concept of word boundaries. The syntax classes in this table aslo determine which characters TEdit thinks are white space (which

gets deleted by control-W along with the preceding word). This can be overridden using the BOUNDTABLE property argument to TEDIT.

TEDIT.DEFAULT.PROPS [Variable]
A default set of PROPS arguments for TEDIT or OPENTEXTSTREAM. Any PROPS the user specifies are APPENDed to a copy of the default. The effect is that any user specifications override the defaults.

1.10 TEdit's Terminal Table and Readtables

When TEdit reads a character from the keyboard, the first thing it does is check to see if it's a command character. TEdit first looks at its default readtable, TEDIT.READTABLE, or at the readtable supplied as the READTABLE property.

Failing that, TEdit then looks to the system terminal table. Characters with terminal sytax-classes CHARDELETE, WORDDELETE, or LINEDELETE act as follows:

CHARDELETE acts as a character-backspace.

WORDDELETE acts like control-W (in fact, this is how control-W is implemented.)

LINEDELETE acts like DEL.

Since the system terminal table is used to implement these functions, you can assign them to other keys at will.

Failing that, TEdit inserts the character at the current insertion point in the document.

The TEdit default readtable is named TEDIT.READTABLE, and it is global. You can use the functions TEDIT.SETSYNTAX and TEDIT.GETSYNTAX to read it and make changes:

(TEDIT.SETSYNTAX CHAR CODE CLASS TABLE) [Function]
Sets the readtable syntax of the character whose charcode is CHAR CODE to be CLASS in the read-table TABLE. The possible syntax classes are listed below.

(TEDIT.GETSYNTAX CHAR CODE TABLE) [Function]
Returns the TEdit syntax class of the character whose charcode is CHAR CODE, according to the read-table TABLE. The possible syntax classes are listed below. An illegal syntax will be returned as NIL.

The allowable syntax classes are:

CHARDELETE Typing this character acts like backspace

WORDDELETE Typing this character acts like control-W

DELETE Typing this character acts like DEL

UNDO Typing this character causes Undo

REDO Typing this character acts like ESC

FN Typing this character calls a specified function (see below)

NONE Typing this character simply inserts it in the document. NIL also has this effect.

You can also cause a keystroke to invoke a function for you. To do so, use the function

(TEDIT.SETFUNCTION CHAR CODE FN TABLE) [Function]
Sets up the TEdit readtable TABLE so that typing the character with charcode CHAR CODE will APPLY FN to the text STREAM and the TEXTOBJ for the document being edited. The function may have arbitrary side-effects.

The abbreviation feature described below is implemented using this function-call facility.

Finally, TEdit uses the read table TEDIT.WORDBOUND.READTABLE to decide where word boundaries are. Whenever two adjacent characters have different syntax classes, there is a word boundary between them. The state of this table can be controlled by the functions

(TEDIT.WORDGET CHAR TABLE) [Function]
Returns the syntax class (a small integer) for a given character. CHAR may be either a character or a charcode; TABLE defaults to TEDIT.WORDBOUND.READTABLE.

(TEDIT.WORDSET CHAR CLASS TABLE) [Function]
Sets the syntax class for a character. Again, CHAR is either a character or a charcode; TABLE defaults to TEDIT.WORDBOUND.READTABLE; CLASS may be either a small integer as returned by TEDIT.WORDGET, or one of the atoms WHITESPACE, TEXT, or PUNCTUATION. Those represent the syntax classes in the default TEDIT.WORDBOUND.READTABLE.

The initial TEDIT.WORDBOUND.READTABLE assigns every character to one of the above classes, along pretty obvious lines. For purposes of control-W, whitespace between the caret and the word being deleted is also removed.

This, too, can be over-ridden for a specific edit session using the BOUNDTABLE property in the call to TEdit.

1.11 The TEdit Abbreviation Facility

The list TEDIT.ABBREVS is a list of ''abbreviations known to TEdit.'' Each element of the list is a dotted pair of two strings. The first is the abbreviation (case does matter), and the second is what the abbreviation expands to. To expand an abbreviation, select it and type control-X. It will be replaced by its expansion.

You can also expand single-character abbreviations while typing. Hitting control-X when no characters are underlined (i.e., after you have typed something) will expand the *single-character* abbreviation to the left of the caret.

Here is a list of the default abbreviations and their expansions:

b The bullet ( )

m The M-dash ( )

n	The gure dash ( )
"	Open double- quotes (‘‘) which can be matched by two normal quotes (’’)