SRI International

E FOR NED USERS

TSC Computer Facility User's Memo No. 1

April 1980 (Revision 10)

David Yost and Judity Westbury
The Rand Corporation
Santa Monica, California

333 Ravenswood Ave. • Menlo Park, CA 94025
(415) 859-6200 • TWX: 910-373-2046 • Telex: 334 486

CONTENTS

Section

I. INTRODUCTION..... 1
Major Changes From Ned..... 2
Document Syntax..... 2

II. RUNNING E..... 4

III. <CMD> KEY..... 5
To Get Off The Command Line..... 7
To Abort A Command..... 7
The Command Mode..... 8

IV. FUNCTION KEYS..... 9

V. <MARK> KEY: FOR DEFINING AREAS WITH THE CURSOR..... 14
How To Mark..... 14
Examples of Marking Commands..... 15
To Cancel Marking..... 16
Describing areas on the Command Line..... 16

VI. RECOVERY..... 18

Appendix

A. SOME EDITING OPERATIONS..... 19
Underlining..... 19
Doublespacing..... 20
Changing text to all upper or lower case..... 20
Using the tee and cat commands from within e..... 21
Invoking other system commands while in e..... 21
Fill, justify, and center..... 22
Replacing text..... 23
Renaming and deleting files..... 25
Windows..... 25
Typing on the Command Line..... 26
Page ejects..... 27

B. ADVANCED AND MISCELLANEOUS TOPICS..... 29
Logging directly into e, bypassing unix shell..... 29
Invoking e and selecting options..... 30
Linked files..... 31
E's work files..... 33
Files and directories..... 35
New commands and other changes..... 36

C. SUMMARY OF FUNCTIONS IN E, LISTED BY TOPIC..... 39

D. SUMMARY OF FUNCTIONS IN E, LISTED BY OLD WAY IN NED..... 49

E. CHART OF <CMD> KEY COMMANDS..... 55

F. CHANGES FROM REVISION 9..... 57

I. INTRODUCTION

First of all, don't be put off by the size of this document. The appendices make up the major portion, and they are mostly tables of information organized for reference.

This paper is intended for users experienced with ned. "E" is a second-generation descendent of "ned", and although it is similar to ned, it incorporates some changes that will totally confound you if you try to use it before reading this. It should be noted that this document has only limited use for those who log directly into E, bypassing the UNIX shell (future Text Processor users).

Ned was put through a metamorphosis to fix all known problems, to make it easier to learn, and to add many new and powerful features for experienced users. What came out of all that is the new editor described here: "E". Perhaps the best news for the user is that recovery after a crash is done automatically for you. The best news architecturally might be that E is virtually unlimited regarding the addition of new commands. Ned relied almost totally on function keys (colored keys) to invoke commands. But the number of function keys is limited, and therefore the number of commands was limited. E, however, has one command key called <CMD> that can process all commands and to which new commands can be added. E also utilizes almost all the function keys too, so you can choose which way you prefer to invoke many commands. E has other advantages, the vast majority of which has been attained without losing the nice things about ned. In fact, much of ned has been retained.

The penultimate appendix of this document lists in some detail

(almost) every command available with E. The last appendix is a chart of commands available with the new <CMD> key.

MAJOR CHANGES FROM NED

E is different from ned in three major ways:

1) the <ARG> (argument) key has now become the <CMD> (command) key;

2) a few of the special function keys (colored keys) are now used differently;

3) the <PUT> key has now become the <MARK> key.

Eventually these particular keys will be physically replaced at your terminal so that, for example, the key that now says ARG on it will say CMD. For the time being, however, just remember that when you press the <ARG> key you will see "CMD:" on your screen.

This document devotes one section to each of these three major changes.

DOCUMENT SYNTAX

For the purposes of this document, the following syntax has been adopted:

1. Capital letters inside "<>" are function keys (colored keys).

   Examples: <RETURN>, <OPEN>, <+PAGE>.

   The one exception to this is the <CTRL> key, or the control key. It is always used in conjunction with another key so its syntax here includes that other key. For example, <CTRL-S> means hold down the <CTRL> key as you press the key with S on it (this will move the text

window over to the right as it always did).

2. <n> means a decimal number.

   Examples: 10, 2, 35.

3. <area> means a specified area within the window in lines or paragraphs and/or columns.

   Examples: "3p" or "3P" (3 paragraphs), "3l" or "3L" (3 lines), "3lx70" or "3Lx70" (a rectangle 3 lines down and 70 columns across), "3px70" or "3Px70" (a rectangle 3 paragraphs down and 70 columns across).

4. <cursor> means any sequence of cursor movements.

5. Other lower case letters inside "<>" mean a word or phrase of that type.

   Example: <filename>.

6. Letters or words inside "[]" indicate options.

   Example: [<area>].

II. RUNNING E

To edit a file, type "e" instead of "ned".

Example:

```
% e <filename>
```

When E calls up a file, it first puts out a message like "e rev 10 is starting...". This lets you know that you can start typing ahead at that point.

If you call up a new file, one that you haven't created before, an empty window will appear and below it E will ask, "Do you want to create <new filename>?" If you do, type the single letter "y" or "Y" and the window will open for your use.

E refrains from clearing the screen until it has to, and does not clear the screen upon exit.

III. <CMD> KEY

The biggest change E incorporates is the <CMD> key (known in the past as the <ARG> key), which can invoke any command. When giving a command with the <CMD> key, the syntax is always the same: Hit the <CMD> key, type out your command(s) with area and/or option(s), then hit the <RETURN> key or other function key:

Example:

```
  <CMD> fill <RETURN>
This will fill one paragraph of text.
As with ned, one paragraph is the default for fill,
justify, and center.
```

When you type this command, you will notice that "CMD:" appears on your screen right below the bottom window line, just as "ARG:" used to. We will refer to that line as the "Command Line". The line directly below the Command Line is now referred to as the Info Line, and that is where the word "INSERT" appears when you press the <INSERT MODE> key. If you use the <MARK> key (more later), the word "MARK" will also appear on the Info Line, next to where "INSERT" appears.

With the <CMD> key, your typed-out command can be abbreviated as long as the abbreviation is unambiguous. While we are on the subject of abbreviations, some comands take options, and they can be abbreviated also. Upper and lower case are not distinguished for command names and their options.

Example:

```
  <CMD> fi <RETURN>
This will also fill one paragraph.
```

If you want to override the default of one paragraph, right after you type out the command, type out the area you wish the command to

work on.

Examples:

<CMD> fi 31 <RETURN>
This will fill three lines down from the cursor.

<CMD> fi 5 <RETURN>
or:
<CMD> fi 5p <RETURN>
This will fill five paragraphs down from the cursor.

Every command invoked by function keys (colored keys) is now also available through the <CMD> key. For example, you can open or close lines as before by hitting the <OPEN> or <CLOSE> keys. But in E you can also type out these commands with the <CMD> key.

Example:

<CMD> open <RETURN>
This will open one blank line.
(As with ned, one line is the default for the
<OPEN>, <CLOSE> and <PICK> function keys and, as above,
when these functions are typed out with the <CMD> key.)

Some commands are no longer available on function keys. You used to exit from ned by hitting the <DEL> key. You can't do that anymore; now you exit via the <CMD> key. (No longer will exit if you accidentally hit the <DEL> key when you intended to hit <RETURN>.)

Example:

<CMD> exit <RETURN>
This will exit you from E and update your files.

Besides the familiar commands in the examples above, E incorporates some entirely new commands to be used with the <CMD> key (see Appendix D).

TO GET OFF THE COMMAND LINE

a) If you hit the <CMD> key and then decide you don't want to give a command, simply hit the <RETURN> key. "CMD:" on your screen will disappear and your cursor will move from the Command Line back into the text window for editing.

b) If you have hit the <CMD> key, typed out a command, and then decided you don't want to give the command, backspace over the typed-out command and then hit <RETURN>.

Or type <CTRL-C>, which will wipe out the Command Line with the command in it, and move your cursor back into the text window. (<CTRL-C> is now the "Interrupt" key. More on that later.)

TO ABORT A COMMAND

You have hit the <CMD> key, typed out your command, and hit <RETURN>, only to change your mind. You want to abort the command as it is working. Hitting <CTRL-C> (the interrupt) will do this. The interrupt works on searches and the following commands: run, feed, fill, justify, and center.

NOTE: Typing within the text window while the command is executing will not automatically abort the command, as was true in the latest ned. Therefore, you no longer have to wait for a command to complete execution before typing ahead.

THE COMMAND MODE

The notion of "Command Mode" has been created. If you give this command:

<CMD> command <RETURN>

your cursor will stay on the Command Line so that you can give continuous commands. When you wish to return your cursor to the text window, type:

<CMD> -command <RETURN>

This facility may be withdrawn later if it is not found useful.

IV. FUNCTION KEYS

Some of the meanings for function keys (colored keys) have been changed or deleted. Examples of this have been noted above, including <CTRL-C>, which used to mean "change windows" but now means "interrupt," and the <DEL> key, which used to exit you from ned but now has no meaning. <DEL> is only biding its time, however; it will be used later, when we get to some of the Features of the Future. The other function key with a big change is <GOTO>. It used to take you to a specified line number, but now it has become the <REPLACE> key. More on that later.

Below is a listing of each function key, what it used to do in your text window, and what it does now under E.

FUNCTION KEY	OLD USE	NEW USE
<ARG>	Introduced arguments for functions.	Is now the <CMD> key (see <CMD> KEY section).
<+TAB> and <-TAB>	Moved the cursor to next tab setting.	Same, except that they don't wrap around any more, and they won't take you all the way to a border unless there is a tabstop there.
<BS>	Backspaced to erase text.	Same, plus: To erase text to left of cursor on cursor line, hit: <CMD> <BS> with INSERT Mode off. To close text to left of cursor on cursor line, hit: <CMD> <BS> with INSERT Mode on.
<DEL>	Exited user from ned.	Has no use now. To exit from E and save your file, type: <CMD> exit <RETURN> To exit and abort the changes you made, type: <CMD> exit abort <RETURN>

<RETURN>	Returned the cursor to first column, next line.	Same, but also executes <CMD> key commands (see <CMD> KEY section).
<HOME>	Moved cursor to upper left-hand corner of text window.	Same, plus: To move cursor to lower left-hand corner, type: <CMD> <HOME>
Arrows	Moved cursor around screen.	Same, plus: Cursor no longer wraps around th window. If you are at the bottom line of a window, and you hit <down arrow> the screen will scroll up. Similarly for the other three directions. You can use <left arrow> and <right arrow> on the Command Line. To move cursor to the end of text on current line, hit: <CMD> <right arrow> If you are beyond end of text on current line, <CMD> <right arrow> takes you to rightmost column. To move cursor to column 1 of current line, hit: <CMD> <left arrow> To move cursor to line 1, same column, type: <CMD> <up arrow> To move cursor to bottom line of window, same column, type: <CMD> <down arrow> (If the end of the file comes before the bottom of the windou you will stop there, and you ca get to the bottom of the windou by typing <CMD> <down arrow> again.
<GOTO>	Moved the text window to the beginning of the file. With <ARG> moved to the end of the file or to a specific line number.	Is now the <REPLACE> key. See section on replacing in Appendix C. To move to the beginning of the file, type: <CMD> goto b <RETURN> or <CMD> <-PAGE>

		To move to the end of the file, type: <CMD> goto e <RETURN> or <CMD> <+PAGE> To move to a certain line number, type: <CMD> goto <n> <RETURN> <CMD> goto <n> <RETURN>
<+PAGE> and <-PAGE>	Moved the text window forward or backward one page.	Same. To move the window forward 4 pages, type: <CMD> 4 <+PAGE> Backward 2 pages would be: <CMD> 2 <-PAGE>
<+LINE> and <-LINE>	Moved the text window forward or backward 11 lines.	Same. To move the window forward 4 lines, type: <CMD> 4 <+LINE> Backward 2 lines would be: <CMD> 2 <-LINE>
<+SCH> and <-SCH>	Searched through text for a given word or string.	Same. Use <CMD> key as you would have used <ARG> before. For example, <CMD> <text> <+SCH> or <CMD> <text> <-SCH>
<INSERT MODE>	Inserted characters or blanks where the cursor sat.	Same. However, when you hit the key, you no longer see "INSERT MODE" on the line below the window. Now you see "INSERT" on the following line (the Info Line). Also, you can use it to edit on the Command Line.
<DEL CHAR>	Deleted characters or blanks where the cursor sat.	Same, plus: You can use it to edit the Command Line. To remove text to right of cursor on current line, hit: <CMD> <DEL CHAR>
<PUT>	Put what was in the pick buffer, or with the <ARG> key put what was in the	Is now the <MARK> key (see <MARK> KEY section). To put the pick buffer, type:

	close buffer.	<CMD> <PICK> or <CMD> -pick <RETURN> To put the close buffer, type: <CMD> <CLOSE> or <CMD> -close <RETURN>
<OPEN>	Inserted a blank line above the line where the cursor sat.	Same. To open 4 lines, type: <CMD> 4 <OPEN> or <CMD> open 4 <RETURN>
<CLOSE>	Closed up the line where the cursor sat.	Same. To close 2 lines, type: <CMD> 2 <CLOSE> or <CMD> close 2 <RETURN>
<PICK>	Placed a line in the pick buffer without removing it from text.	Same Or: <CMD> pick <RETURN> To pick 4 lines, type: <CMD> 4 <PICK> or <CMD> pick 4 <RETURN>
<CTRL-S>	Moved the text window over to the right 16 columns at a time.	Same. To move window over to column where cursor sits, type: <CMD> <CTRL-S>
<CTRL-A>	Moved the text window over to the left 16 columns at a time.	Same. To move window left all the way back to original position, type <CMD> <CTRL-A>
<CTRL-B>	Brought another file into the text window, and exchanged current file and alternate file.	Now this only alternates between current file and alternate file To bring up another file by name, type: <CMD> e <filename> <RETURN>
<CTRL-Z>	Created a new window. With <ARG>, closed most-recently-made window.	Now it moves you from window to window. To create a window, type: <CMD> window [<filename>] <RETU To close last-made window, type: <CMD> -window <RETURN>
<CTRL-C>	Moved you from	Is now the "Interrupt Key"

	window to window.	(see "To Abort A Command" above). To move from window to window, type: <CTRL-Z>
<CTRL-V>	Updated file on disk with present textual changes.	Has no meaning now. You can no longer update your file this way. You can save the file (with its present changes) to another filename by typing: <CMD> save <new filename> <RETL You can save your file in another directory by typing: <CMD> save <pathname> <RETURN>
<CTRL-X>	Executed unix programs when given with the <ARG> key.	Has no meaning now. The most commonly used unix commands are now built-in commands, e.g. "fill". To run a system command, see "EXECUTE" section in Appendix C
<CTRL-[>	Set a new tab where the cursor sat, or with <ARG>, clear a tabstop, or with <ARG> filename, set tabs according to a file of tab settings.	Same, except: To set tabs according to a tab file, type: <CMD> tabfile <file> <RETURN> There's more. See "TABS" sectio in Appendix C.

U. <MARK> KEY: FOR DEFINING AREAS WITH THE CURSOR

Many commands, such as open and close, can operate on a number of lines or a rectangular area as defined by the cursor. The <MARK> key (previously known as the <PUT> key) allows you to "mark" such areas using the cursor. In ned, the <ARG> key was used for marking areas. Marking is much more flexible now that marking is done with a separate <MARK> key because now you can invoke functions requiring the <CMD> key to help you define your area--for example, moving off the current window or go to a certain line number. In fact, you can give just about any <CMD> key command you want while marking.

You should play with marking using a scratch file until you feel comfortable with it.

HOW TO MARK

Hit the <MARK> key; this will mark the current cursor position. On the Info Line under the text window you will see "MARK 1", which means you have marked 1 line. Now you can move the cursor and the window around to mark as many lines or as large a rectangle as you like. You can also use the <CMD> key freely as necessary to do things such as search for strings or go to the end of the file. All the while to the right of the "MARK" indicator you will see numbers noting how big an area you have defined. If you have marked a rectangle, you will see something like "MARK 100x40", meaning 100 lines by 40 columns.

The area you have marked is defined between the current cursor position and the original cursor position (the beginning of your

marked area). To check the boundaries of the marked area, you can move the cursor back and forth between these two positions simply by hitting <MARK> again. This changes only the cursor position; the marked area, as shown by the "MARK" numbers, remains the same. After moving back and forth between the limits of the marked area in this way, you can still continue to redefine the area as before. However, be careful to note the "MARK" indication, always making certain the cursor defines only the area you want.

Now that you have marked the lines or rectangle you want, perform the operation you choose--either by hitting a function key (such as <CLOSE> to close the area you've marked) or by giving a command with the <CMD> key (such as "<CMD> center <RETURN>" to center the lines you've marked). When the command finishes execution the cursor will return to the upper-left corner of the area you marked. If necessary, the window will be moved so that the cursor can be put there. Also, after execution is completed, the marking (e.g., "MARK 100x40") will disappear.

WARNING: Marking can be dangerous. If you forget that you have something marked, you can, for example, close a much bigger area than you intended.

EXAMPLES OF MARKING COMMANDS

Some marking examples are:

<MARK> <cursor> <CLOSE>
This will close the lines or the rectangle the cursor has defined and put the lines or rectangle into the <CLOSE> buffer.

<MARK> <cursor> <OPEN>
This will insert blank lines or a blank rectangle into

the area the cursor has defined.

<MARK> <cursor> <CMD> justify <RETURN>
This will justify the lines the cursor has defined.

<MARK> <CMD> <+PAGE> <CLOSE>
This will close up the lines between the original cursor position and the end of the file, which was moved to with the +PAGE command. It will put the lines in the <CLOSE> buffer.

<MARK> <CMD> goto <n> <RETURN> <CMD> justify <RETURN>
This will justify the lines between the original cursor position and the line number you moved to with the goto command.

TO CANCEL MARKING

If you want to cancel the marking you are doing, type <CMD><MARK> There is a danger here: if you are giving commands via the <CMD> key while marking, you could accidentally cancel the marking.

DESCRIBING AREAS ON THE COMMAND LINE

Rather than using the <MARK> key, you can designate an area by typing its size on the Command Line. To do so, hit the <CMD> key, type out the area you wish to define, then hit the function key you wish.

Examples:

<CMD> 100x40 <CLOSE>
This will close a rectangle 100 lines down by 40 columns across.

<CMD> 1p <CLOSE>
This will close to the end of this paragraph.

<CMD> 3 <OPEN>
This will insert three blank lines.

To designate an area to a typed-out command, hit the <CMD> key, type the command and the area, then hit <RETURN>.

Example:

```
<CMD> center 5L <RETURN>
This will center five lines.
```

VI. RECOVERY

If the system crashed when you were in E, recovering your lost work is simple. When the system comes up again, change into the directory you were in when you started your crashed E session, and type "e" with no arguments whatsoever, including no filename. E will redo your lost session quietly. When it is done, the screen will be updated to the way it looked just before the crash. At that point, you should exit immediately to save your work, just as you would in ned.

Ned used to allow you to watch a fast replay of your crashed session on the screen. E gives you that option. To get a recovery "movie" is not so simple. Refer to the following two sections in Appendix B: "Invoking E and Selecting Options" and "E's Work Files".

Appendix A

SOME EDITING OPERATIONS

Ned allowed you to give special editing commands, such as for doublespacing lines ("space") and sorting ("sort") by sending your text to a system command and replacing that text with the output of the system command.

This capability has been retained and expanded by E.

However, you no longer use <CTRL-X> to execute system commands. Now you use the "run" command with the <CMD> key:

<CMD> run [<area>] <system command> <RETURN>

To see exactly how "run" works, note the examples of special editing commands below, plus some other useful editing operations.

If you don't specify an <area> for the "run" command, the default is zero lines. If you give a number, it will be considered as a number of lines. (This is different from the ned <CTRL-X> which defaulted to 1 paragraph.) Therefore, if you want the command to work on a paragraph you must type "1p" or "1P" as the <area>.

UNDERLINING

You can underline in two ways, as you could before.

a) To underline something short: If you wish to underline one word you can, as you have always been able to, type:

<CTRL-\>H

once for each letter, and then type an underline for each letter.

b) To underline a line of text, move the cursor to that line, and then type the old underline command while invoking "run" with the

<CMD> key.

That is:

<CMD> run 1l just .ul <RETURN>

This will underline the current line.

DOUBLESPACING

If your file is singlespaced and you wish to doublespace or triplespace all or part of it, use the old "space" command with the "run" command. -2 indicates doublespacing and -3 indicates triplespacing. (As before, you cannot singlespace a doublespaced or triplespaced file.)

Some examples:

<CMD> run 5l space -2 <RETURN>
This will doublespace five lines down from the cursor.

<CMD> run 2p space -3 <RETURN>
This will triplespace two paragraphs down from the cursor.

CHANGING TEXT TO ALL UPPER OR LOWER CASE

Once again, this is done as it used to be, except you use the "run" command with the <CMD> key instead of using <CTRL-X>. To convert two paragraphs to upper case, type:

<CMD> run 2p dd conv=ucase <RETURN>
or
<CMD> run 2p tr "[a-z]" "[A-Z]" <RETURN>

To convert one line to lower case, type:

<CMD> run 1 dd conv=lcase <RETURN>
or
<CMD> run 1 tr "[A-Z]" "[a-z]" <RETURN>

This also converts the first letter of every sentence, so you must

change each one back by hand.

USING THE TEE AND CAT COMMANDS FROM WITHIN E

In ned, you might have used the "tee" and "cat" system commands with <CTRL-X> to move text from one file to another. You should not use those commands in this way in E. Their use is inherently tricky and can ruin your chances for a successful replay in the event of a crash. (This was true in ned, too.) Rather, you should edit one file, pick from it the lines you want, then edit the second file and put the lines there. Now that E lets you easily mark as many lines as you like, this should be a simpler way to accomplish the same task anyway.

INVOKING OTHER SYSTEM COMMANDS WHILE IN E

As with ned, you can issue many system commands while you are still in E, inserting their output into the text window. Again, in ned you would use the <CTRL-X> key, but in E you use the "run" command.

Since the default number of lines sent to the command and replaced by its output is zero in E, you can run ordinary system commands like "who" easily.

Some examples:

<CMD> run df <RETURN>
This will list off disk space for all directories; the listing will begin where the cursor sits.

<CMD> run df /mnt <RETURN>
This will list the disk space in /mnt; the information will appear on the cursor line.

<CMD> run ls <RETURN>
This will list the files of the directory you are in.

Another new command similar to "run" is the "feed" command. It is just like the run command, but it doesn't close out the text that is sent to the system command.

WARNING: Since the result of running many commands can be different depending on when you run them, you can ruin your chances for a successful replay in the event of a crash by using this feature. For example, say you run a "who" command and then close out the lines after looking at the result. Then you go on editing, and the system crashes. Now you try to recover, and when the "who" is issued at the time of the replay, there are fewer users on the system, and so there are fewer lines to close. The replay, however, goes on closing the lines as you did in the original session. The replay will have closed some lines it shouldn't have.

FILL, JUSTIFY AND CENTER

Fill, justify, and center are new built-in commands that should never be used with "run". First, invoking them with "run" is inefficient, and second, replay cannot be guaranteed. You should type these commands in the standard <CMD> key format as described in the <CMD> KEY section:

<CMD> ju <RETURN>
This will justify one paragraph.

You can override the default line length of 75 columns by adding a width option of whatever line length you prefer.

Example:

<CMD> ju width=65 <RETURN>
or
<CMD> ju w=65 <RETURN>

```
This will justify one paragraph, giving it a line length
of 65 columns.
```

Whatever width you set becomes the default line length until you redefine it. The same line length will still be in effect if you exit and reenter E with no file arguments.

REPLACING TEXT

In ned you used the "rpl" system command with <CTRL-X>. E has a new "replace" command. Example:

```
<CMD> replace 50 /right/left/ <RETURN>
```

replaces the word "right" with the word "left" throughout the next fifty lines, including the current line starting at the cursor position. Here is the formal specification for the replace command:

```
<CMD> replace [<area>] [show] /<str1>/[<str2>]/ <RETURN>
        or                  or
      -replace            [interactive]
```

where "/" represents the "string delimiter" and can be any printing character except letters, numbers, "." or ",". ("-replace" works backwards in the file.) If no <area> is given, then "all the way in the appropriate direction" is assumed. You can also mark an area for the replace commands. A forward replace starts with the first occurrence of <str1> on or after the current cursor position, and a backwards replace starts with the first occurrence of <str1> to the left of the current cursor position.

<str1> is the "search string", and <str2> is the "replacement string". If <str2> is null, as in "replace /the//" then all

occurrences of the first string are deleted. You are given freedom to pick your own string delimiter so that any character can appear in a search or replacement string.

Normally, a replacement is done quickly, all at once, and the only thing you will see happening on the screen is the replacements that are in your current window position. You can select the "show" option to show you all of the replacements as they are happening, and the window will move as necessary so that you can see them happen.

The "interactive" option sets up the <REPLACE> key (old <GOTO> key) for interactive replacing. Here's what happens: An initial search for <str1> is attempted. If you specified an <area>, then the search is limited to within that area. The <+SEARCH> and <-SEARCH> are set to search for <str1>, and the <REPLACE> key is "armed" to do the replace you specified. If <str1> was found, you can do one of three things:

1. type <+SEARCH> or <-SEARCH> to skip to next instance of <str1>
2. type <REPLACE> to do the replace
3. abandon the replacing and go on to other editing

The <REPLACE> key will remain "armed" until the next "replace" command. You can type it at any time, but it won't do anything unless the cursor is at an instance of <str1>. The <+SEARCH> and <-SEARCH> keys will stay set to search for <str1> until you use them to search for something else.

An "interactive" or "show" option can be either before or after an <area> option on the Command Line.

You can use control characters in either <str1> or <str2> by using the appropriate <CTRL-\><char> sequence (e.g. <CTRL-\>L for form-feed). Newlines (<CTRL-\>J) are a special case, however. They cannot appear in the replacement string, and they can only appear at the beginning and/or end of the search string.

RENAMING AND DELETING FILES

Two other new commands have been added:

name <newname>

delete

Each of these works on the current file. "Name" assigns a new name to that file upon exit, if possible. Delete marks it for deletion upon exit. Nothing really happens until exit, so that your edit session is replayable.

WINDOWS

To create a window, set the cursor where you want the window to appear, then type:

<CMD> window <filename> <RETURN>

where <filename> is the name of the file you want in the new window. The current file will become the alternate file of the new window. If you don't type a filename, the current file and alternate file (if any) will carry over to the new window, so you will have two windows of the same file. (Old ned used to bring up an empty default file.) Since E accepts abbreviations for commands, you could type:

<CMD> w <filename> <RETURN>

and get the same results. You can create up to ten windows, as in

ned.

To move from window to window, type:

<CTRL-Z>

Windows are numbered in the order they were created, so you can go to the third window by typing:

<CMD> 3 <CTRL-Z>

To remove the last window you created, type:

<CMD> -window <RETURN>
or
<CMD> -w <RETURN>

TYPING ON THE COMMAND LINE

E lets you type <CTRL-\> on the Command Line and thus: You can search for strings with control characters in them! You can also use any number of occurrences of <CTRL-\>J or <CTRL-\>j in search strings to specify beginning-of-line or end-of-line context. When used by itself as an argument to <+/-SEARCH>, <CTRL-\>J will find the ends of lines; <CTRL-\>J<CTRL-\>J finds blank lines.

E lets you edit the Command Line much as you would edit a line in the text window. The key to this new facility is that you can use the left and right arrow keys on the Command Line as you would in the text window. All of the other stuff you can do with <INSERT MODE>, <BS>, and <DEL CHAR> are functional on the Command Line, too. Note that once you are on the command line, certain keys like <DEL CHAR> take on an alternate meaning when preceded by the <CMD> key, just as in the text window. For example,

<CMD> <DEL CHAR> will erase the rest of the Command Line.

In addition to these features which work just as they do in the text window, there are two other useful features:

<CMD> <CTRL-B> will bring up the Command Line as it was last time you typed on it, and

<CMD> <PICK> will insert into the Command Line the word pointed to by the cursor in the text window. ("word" here is all characters to the right of the cursor up to the first blank).

The old, easy way to search for a string that already exists in your text still works: Go into the text window and set the cursor at the beginning of the string (a "string" here is all characters to the right of the cursor up to a blank); then hit:

<CMD> <+SCH> This will move the cursor to the next occurrence of the string. It will also put that string into the searchkey buffer and the "last command" buffer.

To execute commands given on the Command Line, the cursor need not be at the end of the typed-out command. It can be anywhere under the command when you type <RETURN>. However, if you are typing <CMD> <string> <+SRCH>, you can only get trailing spaces into the search string by having the cursor be after the last space when you type the <+SRCH> key. (Similarly for <-SRCH>, of course.

PAGE EJECTS

Page ejects are set the same way in E as in ned. Type:

<CTRL-\>L

Appendix B

ADVANCED AND MISCELLANEOUS TOPICS

This section is basically for system programmers, though there may be specialized information here just for you. If you're interested, read on. If you don't understand something just skip it.

LOGGING DIRECTLY INTO E, BYPASSING UNIX SHELL

Some users log directly into E without entering the UNIX shell. To deal with their special situation, the following two new features were put in:

If this user logs on when there is no state file to tell E what he was editing last, E will create a file called "scratch" in the current directory. This file can be used as usual and will be saved on normal exit. (Ned would have put up a default file in that case with a message in it saying there was nothing to edit.)

E login users are given a special way to get a shell to do things difficult or impossible to do from within E. For them the "shell" command has been created. While in E they can type:

<CMD> shell <RETURN>

and this will move them into the shell without logging them off. When the shell exits, they are prompted to hit <RETURN>, which will take them back into E. E is reentered as if it had been invoked with no arguments. The new "call" command works similarly but allows the user to give one system command to be run by the shell before returning to E.

Both of these commands go through the same sequence of updating

files, saving the edit state, restoring tty modes, etc., that E would do on a normal exit, but then they run a shell as a sub-process. Call runs the program with the given arguments. Shell, if invoked with no arguments, runs an interactive shell. Shell with arguments is equivalent to using call.

INVOKING E AND SELECTING OPTIONS

If the system didn't crash while you were in E, invoking it with no arguments (just typing e) brings up all windows and alternate files from the last session, just as "ned !" used to. (If there was a crash, this is the recovery procedure.)

When you invoke E, you can specify some options; they should be typed in this order:

e <options> <filename>

The options are -help, -inplace, and -notracks, -replay, and -silent.

If you type

e -help <other option(s)> [<filename>]

E will list out for you what options are available and put an asterisk next to the ones you just invoked, including defaults.

Typing

e -notracks <filename>

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

To use the -inplace option, see "Linking" below.

Typing:

e -replay

will cause E to replay the last session. When replay is completed,

save by exiting as usual.

Typing:

e -replay=filename

allows you to replay with a specified keys file, where "filename" represents that keys file.

Typing:

e -silent -replay

allows you to replay the recovery, but E won't show you the replay. You will see the file after replay is completed. You can't invoke the -silent option without the -replay option.

If you want to watch a crash recovery, you will use the "-replay" option, which will do the replay from the keys file made by the crashed session. You will have to do one more thing, however, before you can get a crash replay. You will have to delete the E's "changes" work file, i.e. the file it used for the changes during the crashed session. When you do a normal silent recovery by typing e with no arguments, E automatically removes its changes file. The sure way to find out the name of the changes file so you can delete it is to run "e -help". E will tell you the name of the changes file, and you can delete it. See also the section below on "E's Work Files".

LINKED FILES

If you make changes to a file that has multiple links, the normal updating procedure when you exit from E is as follows: your link to the file is renamed to ",<filename>" and the changed version is written out to a completely new file with only one link. Thus you won't disturb the file that everyone was originally linked to.

E gives you the option of updating so that the changes are made in the original file with all of the links. This is called updating "in place". What you need to do is set the "inplace" flag associated with the file(s) you want updated that way. One way to do that is to run your E session with the "-inplace" option. That will make "inplace" the default method of updating all multilink files. Another way is to explicitly set the "inplace" flag for the file(s) with E's new "update" command.

<CMD> update inplace <RETURN> will set the "inplace" flag for the current file.

<CMD> update -inplace <RETURN> will clear the "inplace" flag for the current file.

<CMD> -update <RETURN> will make sure that no matter how many changes you have made or will make to the file, it won't be updated at all on exit.

One more point on links. Say you are editing a file called "f1", and there is another file, "f2", linked to it, which is to say that that file is also known as f2. Now you type <CMD> edit f2 <RETURN>. E will know that f2 is just another name for f1, and what you will get is the exact file you are currently in with all the changes you've made. It will even have the name of your current file rather than of the other linked file.

E'S WORK FILES

Since recovery is now automatic, you needn't know about the work files (keys, state, and change files), but if you're interested read on.

If you have write permission in the current directory, the state, keys, and change files are now in the current directory. If you are the owner of that directory, the files are named ".es1", ".ek1", and ".ec1" respectively. Also, there is now a backup keys file called ".ek1b". During a recovery or a replay from the default keys file, this contains the key strokes from your previous editing session. If you are not the owner of the directory, your login name is appended as in ".ek1.day". If you don't have write permission in the current directory, your work files are in /tmp/etmp/ (like the old "/tmp/nedtmp"), with names (based on your login name) such as k1.day, s1.day, and c1.day .

When E exits normally, it writes out a state file containing information such as which file you were editing, where you were in it, and so on. The state file now saves more information than it used to, including the search string, the state of Insert Mode, the time the session started, and the last "width=n" argument to fill, justify, or center. E uses the state file information to set you up in the same edit environment you were when you exited. In ned, if you took an abort exit the state file would be updated. That's no longer true. As before, "e filename" edits that filename and ignores the state file.

Saving your editing environment in your current directory considerably speeds up entering and exiting E. It also allows you to

go back and forth between different environments depending on the directory you are in, and it makes it easier to guarantee that you are in the right directory to do a reliable replay. As a result of this new policy, and to keep from proliferating useless files, the keystroke file is deleted on normal exit. It is not removed when the system crashes, when you exit with the dump option, or when you exit with the abort option, however. The change file is removed except when the system crashes or when you exit with the dump option.

If you exit with the abort option, the state file is not disturbed and the keys file, as noted above, is not removed. Therefore, if you have a good edit session except for your last command, which somehow destroyed your file, you can recover that session by typing

e -replay

and interrupt the replay before that last command replays. It is recommended that you interrupt with either the interrupt:

<CTRL-C>

or by hitting the <CMD> key.

A revision number is put out as the first two bytes of the state file and keys file. Also a number telling the type of terminal you were at during the session is put out as the second and third bytes of the state file and keys file. E now checks that the revision number and terminal screen size in the state file (if used) and the keys file (if replaying or recovering) match the current revision. If not, E complains and stops.

The keys that you type go through a translation stage before they are written to the keys file, so that the codes that are put into the

keys file tell what you typed regardless of what type of terminal you were on.

An interrupt (<CTRL-C>) that doesn't interrupt anything is not put out to the keys file.

The new commands fill, justify, and center take an optional "width=n" argument, which is remembered in the state file until the next time it is set. Recent versions of the fill and just programs kept a file for each user in /tmp/nedtmp/ that kept the last-used line length. Those files are not used any more.

FILES AND DIRECTORIES

If you try to edit a file that doesn't exist and you answer affirmative to the question as to whether you want to create it, the file is NOT actually created at that time, as it used to be in ned. The file will only be created when and if you do a normal exit which does a permanent save. It used to be that ned created an empty file until you exited, when it put the text in. That's why you had to remove the unsaved file from your directory before recovering. E makes the creating of files replayable, no longer necessitating the removal of unsaved files from your directory before recovering. Also, a zero-sized backup file is no longer created.

Files used to always be created with protection mode "rw-r--r--". Now it is "rw-r--r--" if groupid=1 (which is the case for most users). Otherwise the mode is "rw-rw-r--".

File and directory permission checking for "save" and "edit" is completely rewritten. You can edit directories but you can't modify them. You aren't allowed to edit devices (also known as special

files) or save to them. You aren't allowed to try to save to a directory. You can't modify a file or create a file if you can't write in the directory.

In UNIX Version 7, the "SHELL" environment variable is used to find the appropriate shell for you. In Version 6, if there is a "sh" in the current directory, it is used as the shell. Otherwise "sh" is in your private bin, otherwise it's in "/bin/sh".

NEW COMMANDS AND OTHER CHANGES

The save command can now only be used for saving to files not already held internally by E. "Save" with no argument is now illegal.

If you type an interrupt, but there was nothing to interrupt, E will tell you that. <CTRL-C> interrupts searches and the commands run, feed, fill, just, and center.

The "run" and "feed" commands now always work by invoking the shell, so all of the special characters like ? * and ; you type in arguments to the system command are interpreted by the shell. You should never explicitly invoke the fill, justify and center commands with "run".

Two new commands are:

call <program> [<args>]

shell [<args>]

To see more on them, refer to "Users Who Log Directly Into E, Bypassing the UNIX Shell" above.

If an "edit" or "window" command is given with a filename which is already held internally, the file will come up on the screen at the same window position and cursor position within the window as the last

time you saw it.

<CTRL-B> is now only usable to change from current file to alternate file. Instead of

<CMD> filename <CTRL-B>

type

<CMD> edit <filename> <RETURN>

Ned's <ARG> <CTRL-B> feature was getting naive users into terrible trouble when they invoked it by accident, and it has been withdrawn.

The cursor position after <+/-PAGE> and <+/-LINE> and goto is different. The column position is preserved except for a goto to line 1, when the cursor is put "home". <+/-PAGE> leaves the cursor in the same place on the screen. <+LINE> will leave the cursor at the uppermost line in the window if the line the cursor was on is no longer visible. (Similarly for <-LINE>.)

Appendix C

SUMMARY OF FUNCTIONS IN E, LISTED BY TOPIC

FUNCTION	OLD WAY	NEW WAY
ARGUMENT	<ARG>	<CMD>
ABORT		
Abort and exit	<ARG> a <DEL>	<CMD> ex a <RETURN>
Abort a <MARK> command	Nonexistent.	<CMD> <MARK>
Abort a <CMD> command	Nonexistent.	<CTRL-C>
Get off the Command Line	Nonexistent.	<CMD> <RETURN>
ALTERNATE FILE		
Change text window to alternate file	<CTRL-B>	<CTRL-B> or <CMD> e <RETURN>
CENTER		
Center 1 line	<ARG> 11 center <CTRL-X>	<CMD> ce <RETURN>
Center n lines	<ARG> <n>1 center <CTRL-X>	<CMD> ce <n> <RETURN>
Center n paragraphs	<ARG> center <n> <CTRL-X>	<CMD> ce <n>p <RETURN>
CHANGE		
Change windows	<CTRL-C>	<CTRL-Z>
Change to <n>th created window	<ARG> <n> <CTRL-C>	<CMD> <n> <CTRL-Z>
Change to alternate file	<CTRL-B>	<CTRL-B> or <CMD> e <RETURN>
Change the name of the current file	Nonexistent.	<CMD> n <filename> <RETURN>
CLEAR THE SCREEN	Nonexistent.	<CMD> redraw <RETURN>

AND REDRAW WINDOW TO DELETE SYSTEM MESSAGES		
CLOSE		
Close 1 line	<CLOSE>	<CLOSE> or <CMD> clo <RETURN>
Close 4 lines	<ARG> 4 <CLOSE>	<CMD> 4 <CLOSE> or <CMD> clo 4 <RETURN>
Bring 1 line up to end of another (join 2 lines)	<ARG> <CLOSE>	<CMD> jo <RETURN> or <CMD> -sp <RETURN>
Close lines or rectangle	<ARG> <cursor> <CLOSE>	<MARK> <cursor> <CLOSE> or <CMD> <area> <CLOSE> or <CMD> clo <area> <RETURN>
Close area but don't move rest of text (erase)	Nonexistent.	<CMD> er <area> <RETURN>
COMMAND MODE	Nonexistent.	<CMD> co <RETURN>
CREATE WINDOWS		
Create window with new file in it	<ARG> <filename> <CTRL-Z>	<CMD> w <filename> <RETURN>
Create window with current file in it	<ARG> <curfilename> <CTRL-Z> Didn't take alternate file with it.	<CMD> w <RETURN> Takes alternate file with i
CURSOR		
Move cursor to line 1, column 1	<HOME>	<HOME>
Move cursor to bottom line of of screen, column 1	Nonexistent.	<CMD> <HOME>
Move cursor to end of text on current	Nonexistent.	<CMD> <right arrow>

line		
Move cursor to last column of current line	Nonexistent.	<CMD> <right arrow> (repeat if necessary)
Move cursor to column 1, current line	Nonexistent.	<CMD> <left arrow>
Move cursor to bottom line of screen, cursor column	Nonexistent.	<CMD> <down arrow> (repeat if necessary at end of file)
Move cursor to line 1, current column	Nonexistent.	<CMD> <up arrow>
DEFINE AREA	<ARG> <cursor> <function key>	<MARK> <cursor> <function
DELETE		
Delete current file upon exit	Nonexistent.	<CMD> del <RETURN>
Delete individual characters	<DEL CHAR>	<DEL CHAR>
Delete last window created	<ARG> <CTRL-Z>	<CMD> -w <RETURN>
DUMP EXIT	Nonexistent.	<CMD> ex d <RETURN>
EDIT A FILE		
Bring up a file into text window	<ARG> <filename> <CTRL-B>	<CMD> e <filename> <RETURN>
ERASE AREA	Nonexistent.	<CMD> er <area> <RETURN>
WITHOUT MOVING REST OF TEXT		
EXECUTE		
Execute some function	<ARG> <function> <CTRL-X> or <ARG> <n> <func> <CTRL-X> or <ARG> <n>l <func> <CTRL-X>	<CMD> run 1p <func> <RETURN or <CMD> run <n>p <func> <RETL or <CMD> run <n> <func> <RETUR

Execute some function without closing	<ARG> <n> tee <CTRL-X> <ARG> <CLOSE>	<CMD> feed 1p <func> <RETUR
EXIT		
Exit and save	<DEL>	<CMD> ex <RETURN> or <CMD> bye <RETURN> or <CMD> logoff <RETURN>
Exit and abort	<ARG> a <DEL>	<CMD> ex a <RETURN>
Exit and dump	Nonexistent.	<CMD> ex d <RETURN>
Exit, then execute load	<ARG> <DEL>	<CMD> ex l <RETURN>
"FEED" IS LIKE RUN BUT DOESN'T CLOSE	Nonexistent.	See "EXECUTE" above.
FILE		
Bring up alternate file into text window	<ARG> <filename> <CTRL-B>	<CMD> e <filename> <RETURN>
Change text window to alternate file	<CTRL-B>	<CTRL-B> or <CMD> e <RETURN>
Create window with new file in it	<ARG> <filename> <CTRL-Z>	<CMD> w <filename> <RETURN>
Change windows	<CTRL-C>	<CTRL-Z>
Change to <n>th created window	<ARG> <n> <CTRL-C>	<CMD> <n> <CTRL-Z>
Change the name of the current file	Nonexistent.	<CMD> n <filename> <RETURN>
Create window with current file in it	<ARG> <filename> <CTRL-Z> Didn't take alternate file with it.	<CMD> w <RETURN> Takes alternate file with i
Delete current	Nonexistent.	<CMD> del <RETURN>

file upon exit		
Delete last window created	<ARG> <CTRL-Z>	<CMD> -w <RETURN>
Exit and save	<DEL>	<CMD> ex <RETURN> or <CMD> bye <RETURN> or <CMD> logoff <RETURN>
Exit and abort	<ARG> a <DEL>	<CMD> ex a <RETURN>
Exit and dump	Nonexistent.	<CMD> ex d <RETURN>
Exit, then execute load	<ARG> <DEL>	<CMD> ex l <RETURN>
Save current changes onto backup file	<CTRL-U>	No equivalent now.
Save current changes as a second file	Nonexistent.	<CMD> sa <new filename> <RI
Save file in another directory	<ARG> <pathname> <CTRL-U>	<CMD> sa <pathname> <RETURI
FILL		
Fill n lines	<ARG> <n>l fill <CTRL-X>	<CMD> fi <n>l <RETURN> or <MARK> <cursor> <CMD> fi <F
Fill 1 paragraph	<ARG> fill <CTRL-X>	<CMD> fi <RETURN>
Fill n paragraphs	<ARG> <n> fill <CTRL-X>	<CMD> fi <n> <RETURN>
Fill 3 paragraphs overriding default line length	<ARG> 3 fill ".ll 65" <CTRL-X>	<CMD> fi 3 w=65 <RETURN> or <CMD> fi w=65 3 <RETURN> or <MARK> <cursor> <CMD> fi w= <RETUR
GOTO		
Goto beginning	<GOTO>	<CMD> <-PAGE>

of file		or <CMD> g <RETURN> or <CMD> g b <RETURN>
Goto end of file	<ARG> <GOTO>	<CMD> <+PAGE> or <CMD> g e <RETURN>
Goto specific line number	<ARG> <n> <GOTO>	<CMD> g <n> <RETURN>
INPLACE UPDATING		
Set inplace update mode	Nonexistent	<CMD> u i <RETURN>
Clear inplace update mode	Nonexistent	<CMD> u -i <RETURN>
INSERT INDIVIDUAL CHARACTERS	<INSERT MODE>	<INSERT MODE>
JUSTIFY		
Justify n lines	<ARG> <n>l just <CTRL-X>	<CMD> ju <n>l <RETURN> or <MARK> <cursor> <CMD> ju <R
Justify 1 paragraph	<ARG> just <CTRL-X>	<CMD> ju <RETURN>
Justify n paragraphs	<ARG> <n> just <CTRL-X>	<CMD> ju <n> <RETURN>
Justify 3 paragraphs overriding default line length	<ARG> 3 just ".ll 65" <CTRL-X>	<CMD> ju 3 w=65 <RETURN> or <CMD> ju w=65 3 <RETURN> or <MARK> <cursor> <CMD> ju w= <RETUR
MARK AREA WITH CURSOR	<ARG> <cursor> <function key>	<MARK> <cursor> <function k
MOVE TEXT WINDOW		
Move text window to the right	<CTRL-S>	<CTRL-S>
Move text window to the left	<CTRL-A>	<CTRL-A>

Move right so that cursor column becomes left column	Nonexistent.	<CMD> <CTRL-S>
Move all the way back to the left	Nonexistent.	<CMD> <CTRL-A>
OPEN		
Open 1 line	<OPEN>	<OPEN> or <CMD> o <RETURN>
Open n lines	<ARG> <n> <OPEN>	<CMD> <n> <OPEN> or <CMD> o <n> <RETURN>
Open rectangle	<ARG> <cursor> <OPEN>	<MARK> <cursor> <OPEN> or <CMD> <area> <OPEN> or <CMD> o <area> <RETURN>
Open a line in the middle (split 1 line in 2)	<ARG> <OPEN>	<CMD> sp <RETURN> or <CMD> -j <RETURN>
PICK		
Place 1 line in pick buffer	<PICK>	<PICK> or <CMD> pi <RETURN>
Place 4 lines in pick buffer	<ARG> 4 <PICK>	<CMD> 4 <PICK> or <CMD> pi 4 <RETURN>
Place lines or rectangle in pick buffer	<ARG> <cursor> <PICK>	<MARK> <cursor> <PICK> or <CMD> <area> <PICK> or <CMD> pi <area> <RETURN>
PUT		
Put pick buffer	<PUT>	<CMD> <PICK> or <CMD> -p <RETURN>
Put close buffer	<PUT> <CLOSE>	<CMD> <CLOSE> or <CMD> -c <RETURN>

Put erase buffer	Nonexistent.	<CMD> -e <RETURN>
REDRAW THE SCREEN CLEARING SYSTEM MESSAGES	Nonexistent.	<CMD> red <RETURN>
"RUN" EXECUTES COMMANDS	<CTRL-X>	<CMD> run [<area>] <commanc [options] <RETURN
SAVE		
Save current changes onto backup file	<CTRL-V>	No equivalent now.
Save current changes as a second file	Nonexistent.	<CMD> sa <new filename> <RE
Save file in another directory	<ARG> <pathname> <CTRL-V>	<CMD> sa <pathname> <RETURN
Ensure update of current file on exit	Nonexistent	<CMD> u <RETURN>
Prevent update of current file on exit	Nonexistent	<CMD> -u <RETURN>
Set inplace update mode	Nonexistent	<CMD> u i <RETURN>
Clear inplace update mode	Nonexistent	<CMD> u -i <RETURN>
TAB		
Set a tabstop	<S/R TAB>	<S/R TAB> or <CMD> tab <RETURN> or <CMD> tab <column#> <RETURN
Clear a tabstop	<S/R TAB>	<CMD> <S/R TAB> or <CMD> -tab <RETURN> or <CMD> -tab <column#> <RETUR
Set tabs according to a file of	<ARG> <file> <S/R TAB>	<CMD> tabfile <file> <RETUR

tabstops		
Set tabs every <n> columns	<ARG> <n> <S/R TAB> worked for some <n>'s	<CMD> tabs <n> <RETURN>
Set tabs every <n> columns within a range of columns	Nonexistent	<MARK> <right or left arrou <CMD> tabs <n> <RETURN>
Clear tabs on every <n> columns	Nonexistent	<CMD> -tabs <n> <RETURN>
Clear tabs every <n> columns within a range of columns	Nonexistent	<MARK> <right or left arrou <CMD> -tabs <n> <RETURN>
Clear all tabstops	Nonexistent	<CMD> -tabs <RETURN>
Clear all tabstops within a range of columns	Nonexistent	<MARK> <right or left arrou <CMD> -tabs <RETURN>
TEE	<ARG><n>1 tee <name><CTRL-X>	<CMD> <n> <PICK> <CMD> e <name> <CMD> <PICK>
UPDATE		
Ensure update of current file on exit	Nonexistent	<CMD> u <RETURN>
Prevent update of current file on exit	Nonexistent	<CMD> -u <RETURN>
Set inplace update mode	Nonexistent	<CMD> u i <RETURN>
Clear inplace update mode	Nonexistent	<CMD> u -i <RETURN>
WINDOWS		
Create window with new file in it	<ARG> <filename> <CTRL-Z>	<CMD> w <filename> <RETURN>

Create window with current file in it	<ARG> <filename> <CTRL-Z> Didn't take alternate file with it.	<CMD> w <RETURN> Takes alternate file with
Change windows	<CTRL-C>	<CTRL-Z>
Change to <n>th created window	<ARG> <n> <CTRL-C>	<CMD> <n> <CTRL-Z>
Delete last window created	<ARG> <CTRL-Z>	<CMD> -w <RETURN>

Appendix D

SUMMARY OF FUNCTIONS IN E, LISTED BY OLD WAY IN NED

OLD WAY	FUNCTION	NEW WAY
<ARG>	To enter the Command Line	<CMD>
<ARG> <ARG>	To get off the Command Line	<CMD> <RETURN>
<ARG> <command> <ARG>	To get off the Command Line	<CMD> <command> <CTRL-C>
<ARG> 5 <right arrow>	Move the cursor 5 columns to the right	No equivalent.
<ARG> 5 <left arrow>	Move the cursor 5 columns to the left	No equivalent.
<ARG> 5 <up arrow>	Move the cursor 5 lines up	<CMD> 5 <up arrow>
<ARG> 5 <down arrow>	Move the cursor 5 lines down	<CMD> 5 <down arrow>
<ARG> 11 center <CTRL-X>	Center 1 line	<CMD> ce 1 <RETURN>
<ARG> <n>1 center <CTRL-X>	Center n lines	<CMD> ce <n> <RETURN>
<ARG> center <CTRL-X>	Center 1 paragraph	<CMD> ce 1p <RETURN>
<CLOSE>	Close a line	<CLOSE> or <CMD> clo <RETURN>
<ARG> <n> <CLOSE>	Close n lines	<CMD> <n> <CLOSE> or <CMD> clo <n> <RETURN>
<ARG> <CLOSE>	Bring 1 line up to end of another (join 2 lines)	<CMD> jo <RETURN> or <CMD> -sp <RETURN>
<ARG> <cursor> <CLOSE>	Close lines or rectangle	<MARK> <cursor> <CLOSE> or <CMD> <area> <CLOSE> or

		<CMD> clo <area> <RETURN>
<CTRL-A>	Move text window to the left	<CTRL-A>
<CTRL-B>	Change text window to alternate file	<CTRL-B> or <CMD> e <RETURN>
<ARG> <CTRL-B>	Bring up filename at cursor into text window	<CMD> e <CMD> <PICK> <RETUR
<ARG> <filename> <CTRL-B>	Bring up file into text window	<CMD> e <filename> <RETURN
<CTRL-C>	Change windows	<CTRL-Z>
<ARG> <n> <CTRL-C>	Change to <n>th created window	<CMD> <n> <CTRL-Z>
<CTRL-S>	Move text window to the right	<CTRL-S>
<CTRL-U>	Update current changes into current	No equivalent now.
<ARG> <pathname> <CTRL-U>	Save file in another directory	<CMD> sa <pathname> <RETURN
<ARG> <CTRL-Z>	Delete last window created	<CMD> -w <RETURN>
<ARG> <filename> <CTRL-Z>	Create window with new file in it	<CMD> w <filename> <RETURN>
<ARG> <filename> <CTRL-Z>	Create window with current file in it	<CMD> w <RETURN> Takes alternate file with i
<ARG> <cursor> <func key>	Define area with cursor	<MARK> <cursor> <function k

<ARG> <cursor> <ARG>	Abort a defined area	<MARK> <cursor> <CMD> <MARK
<CTRL-X>	Execute commands	<CMD> run [<area>] <commanc <RETUR
<ARG> <function> <CTRL-X> or <ARG> <n> <func> <CTRL-X> or <ARG> <n>l <func> <CTRL-X>	Execute some function	<CMD> run 1p <func> <RETURN or <CMD> run <n>p <func> <RETU or <CMD> run <n> <func> <RETUR
<DEL>	Exit and save	<CMD> ex <RETURN> or <CMD> bye <RETURN> or <CMD> logoff <RETURN>
<ARG> <DEL>	Exit, then execute load	<CMD> ex 1 <RETURN>
<ARG> a <DEL>	Exit and abort changes	<CMD> ex a <RETURN>
<DEL CHAR>	Delete individual characters	<DEL CHAR>
<ARG> fill <CTRL-X>	Fill 1 paragraph	<CMD> fi <RETURN>
<ARG> <n> fill <CTRL-X>	Fill n paragraphs	<CMD> fi <n> <RETURN>
<ARG> <n>l fill <CTRL-X>	Fill n lines	<CMD> fi <n>l <RETURN> or <MARK> <cursor> <CMD> fi <R
<ARG> 3 fill ".ll 65 <CTRL-X>	Fill 3 paragraphs overriding default line length	<CMD> fi 3 w=65 <RETURN> or <CMD> fi w=65 3 <RETURN> or <MARK> <cursor> <CMD> fi w= <RETUR
<GOTO>	Goto beginning of file	<CMD> <-PAGE> or <CMD> g <RETURN> or <CMD> g b <RETURN>
<ARG> <GOTO>	Goto end of file	<CMD> <+PAGE> or

		<CMD> g e <RETURN>
<ARG> <n> <GOTO>	Goto specific line number	<CMD> g <n> <RETURN>
<HOME>	Move cursor to line 1, column 1	<HOME>
<INSERT MODE>	Insert individual characters	<INSERT MODE>
<ARG> just <CTRL-X>	Justify 1 paragraph	<CMD> ju <RETURN>
<ARG> <n> just <CTRL-X>	Justify n paragraphs	<CMD> ju <n> <RETURN>
<ARG> <n>l just <CTRL-X>	Justify n lines	<CMD> ju <n>l <RETURN> or <MARK> <cursor> <CMD> ju <F
<ARG> 3 just ".ll 65" <CTRL-X>	Justify 3 paragraphs overriding default line length	<CMD> ju 3 w=65 <RETURN> or <CMD> ju w=65 3 <RETURN> or <MARK> <cursor> <CMD> ju w= <RETUR
<OPEN>	Open 1 line	<OPEN>
<ARG> <n> <OPEN>	Open n lines	<CMD> <n> <OPEN> or <CMD> o <n> <RETURN>
<ARG> <cursor> <OPEN>	Open rectangle	<MARK> <cursor> <OPEN> or <CMD> <area> <OPEN> or <CMD> o <area> <RETURN>
<ARG> <OPEN>	Open a line in the middle (split 1 line in 2)	<CMD> sp <RETURN> or <CMD> -j <RETURN>
<PICK>	Place 1 line in pick buffer	<PICK> or <CMD> pi <RETURN>
<ARG> 4 <PICK>	Place 4 lines	<CMD> 4 <PICK>

	in pick buffer	or <CMD> pi 4 <RETURN>
<ARG> <cursor> <PICK>	Place lines or rectangle in pick buffer	<MARK> <cursor> <PICK> or <CMD> <area> <PICK> or <CMD> pi <area> <RETURN>
<PUT>	Put pick buffer	<CMD> <PICK> or <CMD> -p <RETURN>
<PUT> <CLOSE>	Put close buffer	<CMD> <CLOSE> or <CMD> -c <RETURN>
<S/R TAB>	Set a tabstop	<S/R TAB> or <CMD> tab <RETURN> or <CMD> tab <column#> <RETURN
<S/R TAB>	Clear a tabstop	<CMD> <S/R TAB> or <CMD> -tab <RETURN> or <CMD> -tab <column#> <RETUR
<ARG> <file> <S/R TAB>	Set tabs according to a file of tabstops	<CMD> tabfile <file> <RETUR
<ARG> <n> <S/R TAB> (worked for some <n>'s)	Set tabs every <n> columns	<CMD> tabs <n> <RETURN>
<ARG> <n>1 tee <new filename> <CTRL-X>	Tee, create new file from old without closing	<CMD> <n> <PICK> <CMD> e <new filename> <CMD> <PICK>

Appendix E

CHART OF <CMD> KEY COMMANDS

Below is a list of the words now recognized as commands by the <CMD> key. Acceptable abbreviations are indicated with underlining. In some cases, the number of required characters is more than you might expect. That is because we already know what some of the future commands will be, and we don't want you to have to relearn the abbreviations when they are introduced. Upper and lower case are accepted for commands and command options. An asterisk denotes something you couldn't do before.

bye	Same as exit (see below).
*call <command>	Users who log into E can give a shell command.
center	Centers area you marked or typed out.
close	Closes up area and puts it in "close" buffer.
*command	Puts you into Command Mode.
*delete	Deletes current file upon exit.
e [<filename>]	Guaranteed unambiguous synonym for "edit"
edit <filename>	Calls up <filename> into text window.
edit	Alternates between files.
*erase	Erases area but does not move rest of text.
exit	Exits you from E and updates files.
exit abort	Exits without updating.
exit dump	Simulates crash; for editor testing.
exit load	Exits and compiles; for software development.
*feed	Like run but doesn't close.
fill	Fills text for rough right margin.
goto	Moves text window to beginning of file.
goto b	Moves text window to beginning of file.
goto e	Moves text window to end of file.
goto <line#>	Moves cursor to specific line number.
join	Joins current line with line below at cursor.
justify	Justifies text for even right margin.
logoff	Same as exit.
name <newname>	Changes name of current file upon exit.
open	Puts blank area into area you marked or typed.
pick	Picks area and puts in "pick" buffer.
*redraw	Redraws window, eliminating system messages.
*replace	Replaces words or strings throughout text.
run	Executes system commands.
save <filename>	Saves current file to <filename>.
*shell	Run a shell as a sub-process.
split	Splits line at cursor.
tab	Set tab(s).
tabfile <filename>	Set tabs according to tab file.
*tabs <interval#>	Set tabs every <interval#> columns.
*update	Ensure updating of current file on exit.
*update inplace	Set inplace update flag for current file.
*update -inplace	Clear inplace update flag for current file.
window	Makes a new window into current file

window <filename>	Creates window of specified file.
-close	Puts "close" buffer.
*-command	Moves cursor from Command Line into text window.
*-erase	Puts "erase" buffer.
-join	Same as split.
-pick	Puts "pick" buffer.
*-replace	Replace text.
-split	Same as join.
-tab	Remove tab
*-tabs	Remove tabs
*-update	Prevent update of current file on exit.
=window	Closes up last window created.

Appendix F

CHANGES FROM REVISION 9

The editor now looks ahead for <CTRL-C> from the terminal, so that if you are doing a search, then type some keys other than <CTRL-C>, then type a <CTRL-C>, the search will be interrupted and all the keys you typed between the search and the <CTRL-C> are thrown away. E9 would have interrupted only the function immediately preceding the <CTRL-C> in the keystroke stream.

The remaining changes in E10 over E9 are related to the handling of terminals, and the normal user need not read further.

Earlier versions of ned and E wrote keys out to the keys file one key behind. Thus the keystroke that caused an editor crash was never in the keys file, and couldn't be read in on a replay or recovery to crash you again. E10 doesn't do this write-behind to the keys file. Instead, it refrains from replaying the last keystroke in the keys file so that you won't crash again.

More than one kind of terminal can be handled by the same copy of the editor. This is handled by compiled-in code in the editor, and presently works only on version 7 unix systems using the environment variable "TERM". Presently handled types of terminals and the values for TERM to operate with them are:

TERM	Terminal type
aa	Ann Arbor 40x80 terminal
3a	Lear Siegler ADM3a
31	Lear Siegler ADM31
k1	Heathkit H89