Full Screen Text Editor
Tutorial

Version 3.10
March 19, 1989

by

Brian Straight and Daniel M. Lawrence

You are looking at the MicroEMACS tutorial. Comments on this document maybe referred to Daniel Lawrence.

NOTE: This tutorial attempts to help you "learn by doing". The characters ">>" at the left margin of your screen indicate directions for you to try using a command.

EMACS commands generally involve the CONTROL key (sometimes labelled CTRL or CTL) or the META key (generally labelled ESCAPE). Rather than write out CONTROL or META each time we want you to prefix a character, we'll use the following abbreviations:

^<chr> Hold the CONTROL key while pressing the character <chr>. Thus, ^F would be: hold the CONTROL key and press F.

>> Now type ^V (View Next Screen) to move to the next screen. Remember: hold the CONTROL key and press V.

ESC-<chr> Press the ESCAPE key and release it, then press the character <chr>. Note: The command will have the same meaning for upper or lower case characters (<chr>).

IMPORTANT NOTE: If you must exit at some point, type ^X^C.

For the time being, you'll be expected to type ^V whenever you finish reading the current screen.

Note that there is an overlap when going from screen to screen; this provides some continuity when moving through the file.

The first thing that you need to know is how to move around from place to place in the file. You already know how to move forward a screen with ^V. To move back a screen, type ^Z.

>> Try typing ^Z and then ^V to move back and forth between screens a few times.

SUMMARY

The following commands are useful for viewing screens:

^V	Move forward one screen
^Z	Move back one screen

ESC-^L Clear screen and redisplay everything, putting the text near the cursor at the center of the screen.

>> Find the cursor and remember what text is near it. Type an ESC-^L. Find the cursor again and see what text is near it now.

BASIC CURSOR CONTROL

Getting from screen to screen is useful, but how do you reposition yourself within a given screen to a specific place? There are several ways you can do this. One way (not the best, but the most basic) is to use the commands previous, backward, forward and next. As you can imagine these commands(which are given to EMACS as ^P, ^B, ^F, and ^N respectively) move the cursor from where it currently is to a new place in the given direction.Here, in a more graphical form, are the commands:

```
                    Previous line, ^P
                           :
                           :
Backward, ^B .... Current cursor position .... Forward, ^F
                           :
                           :
                    Next line, ^N
```

You'll probably find it easy to think of these by letter. P for previous,N for next, B for backward and F for forward. These are the basic cursor positioning commands and you'll be using them ALL the time so it would be of great benefit if you learn them now.

>> Do a few ^N's to bring the cursor down to this line.

>> Move into the line with ^F's and then up with several ^P's. Note what ^P does when the cursor is in the middle of the line.

>> Try ^B at the beginning of a line. Note what happened to the cursor. Do a few more ^B's. Then do ^F's back to the end of the line and beyond.

When you go off the top or bottom of the screen, the text beyond the edge is shifted onto the screen so that your instructions can be carried out while keeping the cursor on the

screen.
>> Move the cursor off the bottom of the screen with ^N's and see what happens. Note the new position of the cursor.

If moving by characters is too slow, you can move by words. ESC-F moves forward a word and ESC-B moves back a word.

>> Type a few ESC-F's and ESC-B's. Intersperse them with ^F's and ^B's.

Notice the parallel between ^F and ^B on the one hand, and ESC-F and ESC-B on the other hand. Very often META characters are used for operations related to English text whereas CONTROL characters operate on the basic textual units that are independent of what you are editing (characters, lines, etc.).

Two other commands which are useful are ^A and ^E. These commands move the cursor to the beginning (^A) and the end (^E) of the line.

>> Try a couple of ^A's, and then a couple of ^E's. Note that the cursor does not move when either of these commands is repeated continuously.

Two other simple cursor motion commands are ESC-< (less than), which moves to the beginning of the file, and ESC-> (greater than), which moves to the end of the file. If you need the shift key to type a "<", then you must also use the shift key to type ESC-<. Otherwise, you would be typing ESC-, .

The location of the cursor within the text is also called "point". To paraphrase, the cursor shows on the screen where point is located in the text.

Here is a summary of simple moving operations, including the word and line moving commands:

^F	Move forward a character
^B	Move back a character
ESC-F	Move forward a word
ESC-B	Move back a word
^N	Move to next line
^P	Move to previous line
ESC-N	Move to next paragraph
ESC-P	Move to

previous paragraph

^A	Move to beginning of line
^E	Move to end of line
ESC-<	Go to beginning of file
ESC->	Go to end of file

>> Try all of these commands now a few times for practice as these are the most often used commands. Since the last two will take you away from this screen, use ^V's and ^Z's to return here.

Like all other commands in EMACS, these commands can be given arguments which cause them to be executed repeatedly. The way you give a command a repeat count is by pressing META (ESC) and then the number before you enter the command. As a special case, typing ^U is equivalent to ESC-4.

For instance, ESC-8 ^F moves forward eight characters.

>> Try giving a suitable argument to ^N or ^P to come as close as you can to this line in one jump.

This also applies to the screen moving commands, ^V and ^Z. When given an argument, they scroll the screen up or down by that many screens.

>> Try typing ESC-3 ^V now.

If you would like to scroll up, you can give an argument to ^Z.

ABORTING COMMANDS

The EMACS command used to abort any command which requests input is^G. For example, you can use ^G to discard a numeric argument or at the beginning of a command that you don't want to finish.

>> Type ESC-100 to make a numeric argument of 100, then type ^G. Now type ^F. How many characters does it move? If you have typed an ESC by mistake, you can get rid of it with ^G^G.

ERRORS

Sometimes you may do something which EMACS doesn't allow. If it is something simple, such as typing a CONTROL key sequence which is not associated with any command, EMACS will just beep at you. Otherwise,EMACS will also display an informative error message at the bottom of
the screen.

Some versions of EMACS do not have all the features described in this tutorial implemented yet. If you come across such an unimplemented feature, you may get an error message when you try to use it. Just press any cursor movement key and proceed to the next section of the tutorial.

NOTE: Several of the exercises in the following sections allow you to use options which will make changes to this tutorial. Do no worry about these changes affecting the tutorial - this is only a copy of the master tutorial and you will not be instructed to save the changes made to it.

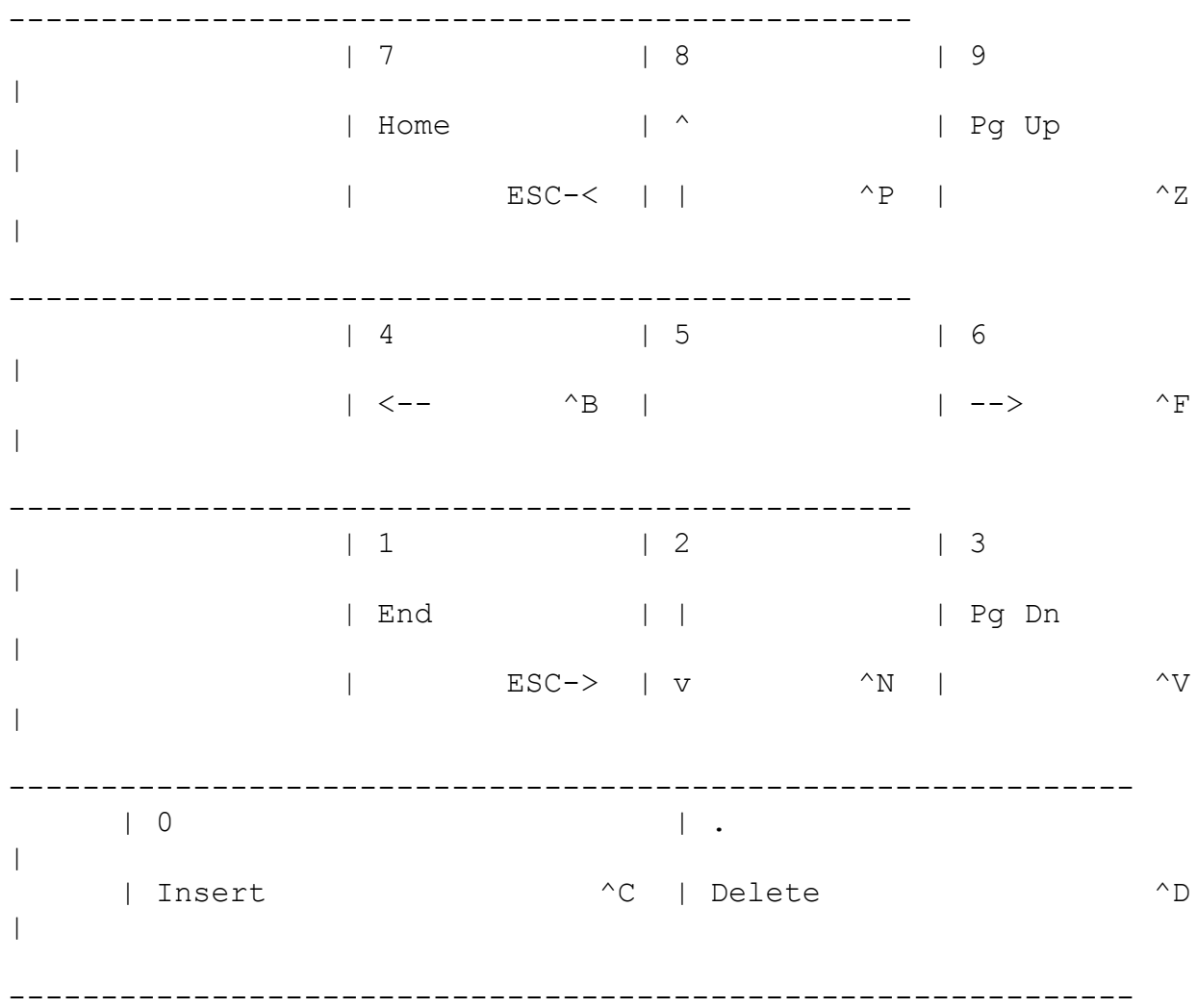
CURSOR KEYS

The cursor keypad, usually located on the right side of the keyboard,has been bound to some of the more useful screen movement commands.The mappings are as follows:

Cursor-Right	^F	Move forward a character
Cursor-Left	^B	Move back a character
^Cursor-Right	ESC-F	Move forward a word
^Cursor-Left	ESC-B	Move back a word
Cursor-Down	^N	Move to next line
Cursor-Up	^P	Move to previous line
Pg-Dn	^V	Move to next screen
Pg-Up	^Z	Move to previous screen

Home	ESC-<	Go to beginning of file
End	ESC->	Go to end of file
Insert	^C	Insert single space
Delete	^D	Delete current character

A map of the keypad layout looks something like this:



>> Practice using the cursor keypad.

MODE LINE

The line above the function key display at the bottom of

the screen is referred to as the "communication line". This is where EMACS interactively communicates with you. Later you will see how EMACS prompts you for information on this line, such as to initiate a search. EMACS can report things to you on this line as well.

>> Type ^X= and see what appears in the communication line. Don't worry about what all this information means - it is just an example of how EMACS lets you know more about the file you are editing.

The line immediately above the communication line is referred to as the"mode line". The mode line looks something like

=*== MicroEMACS 3.10 () == emacs.tut == File: emacs.tut===

This is a very useful "information" line.

- The asterisk (star) indicates that changes have beenmade to the file. Immediately after opening or saving a file, there is no star.

- Any words inside the parentheses indicate the "modes" EMACS is currently in. Modes will be discussed in the next section.

- The string following the () is the buffer name, i.e., the name EMACS gives to the buffer, and it is usually related to the filename.

- The string following "File:" is the name of the file you are currently editing.

>> Look at the mode line and identify the items discussed above.

MODES

Listed within the parentheses are the "modes" which are associated with the current buffer. Modes are a feature of EMACS which assist in the editing of different languages, i.e., C, and text. Presently, there are no modes associated with this buffer. This means EMACS will do exactly what you think it will when using it - no "bonuses". You can find out more about the current buffer and mode status by typing ^X^B. Refer to the

EMACS manual for a further discussion of buffers and modes.

As you become more familiar with EMACS and the use of buffers, "mode"takes on additional meaning. When more than one buffer is in use, a mode is referred to as "local" or "global". These terms indicate how amode will affect the current buffer and other existing or to be added buffers.

A "local" mode is valid only within the scope of the current buffer. Other existing buffers and buffers which will be added are not affected by local modes.

The commands to add and delete local modes are

^XM	Add a local mode
^X^M	Delete a local mode

Each of the above commands will prompt you for a mode. To activate(deactivate) a mode, type the name of a valid (active) mode (refer to EMACS manual for a complete list of the valid modes) and follow it by pressing <Return>, thecarriage-return key.

>> Type ^XM WRAP - note the change in the mode line. Move the cursor to a blank line on this screen and begin typing the sequence "asdf ". Continue typing this sequence and note what happens when the right margin is encountered.

The previous exercise allowed you to enter text with the "WRAP" mode active. As you can see, "WRAP" instructs EMACS to break between words when a line gets too long. However, in order for this mode to be effective, spaces must be inserted between words.

The right margin is usually set at 72 characters but it can be changed.To change the margin type ESC nn ^XF where "nn" is the column number of the new right-hand margin.

>> Type ESC 40 ^XF. Then begin typing "asdf " and notice where the line now breaks. To return to the default right-hand margin, type ESC 72 ^XF.

>> Type ^X^M WRAP to "turn off" the local mode "WRAP".

A "global" mode affects only those buffers which will be ADDED after the"add/delete global mode" command is executed - not the current or other existing buffers. Currently there is

no global mode set.

The commands to add and delete global modes are

ESC-M	Add a global mode
ESC-^M	Delete a global mode

Note: All modes can be local. However, global modes allow you to activate those modes which usually apply to most of the buffers in use.

As with local modes, each of the above commands will prompt you fora mode. To activate (deactivate) a mode, enter the name of a valid(active) mode.

>> Type ESC-M OVER. This mode tells EMACS to write over the text on the current line. Is there any change in the mode line? Now move to the line of "asdf " you entered and start typing. Note that nothing happens. Remember that global modes affect only those modes which will be added - not those already existing.

>> Type ESC-^M OVER to "turn off" the global overwrite mode.

INSERTING AND DELETING

If you want to type text, just start typing. Characters which you can see, such as A, 7, *, etc. are taken by EMACS as text and are immediately inserted. Type <Return> to insert a line separator,i.e., a single linefeed character.

You can delete the last character you typed by typing either <Delete>or ^H. On some keyboards, there is a dedicated key for creating a ^H.If so, it is usually labelled as either "Backspace" or "<--". <Delete>is a key on the keyboard, which may be labelled "Rubout" instead of "Delete" on some terminals. More generally, <Delete> deletes the character immediately before the current cursor position.

>> Now type a few characters and then delete them by typing <Delete> a few times.

>> Now start typing text until you reach the right margin, then continue to type. When a line of text gets too big for one line on the screen, the line of text is "continued" off the edge of the screen. The dollar sign at the right margin indicates a

line which has been continued. EMACS scrolls the line over so you can see what you are editing. The "$" at the left or right edge of the screen indicates that the current line extends off in that direction.

This concept is easier to understand by doing rather than by reading about it so it is suggested that the following exercises be done.

>> The following line actually goes off the edge. Try typing enough ESC-F's so that you move off the right hand end of this line. This is a long line of text. Note the "$" at each edge. Keep typing ESC-F's and watch where EMACS decides to scroll the line. Now, type ESC-B's until EMACS decides to scroll the line again.

>> Go to the line you entered which the text continued off the edge of the screen. Use ^D's to delete the text until the text line fits on one screen line again. The continuation "$" will go away.

>> Move the cursor to the beginning of a line and type <Delete>. This deletes the line separator before the line and merges the line onto the previous line. The resulting line may be too long to fit on the screen, in which case it has a continuation indicator.
>> Press <Return> to insert the separator again.

Internally, EMACS will allow you to have lines of nearly any length, limited only by the amount of memory available. Externally, however, EMACS can only read or write lines, to or from a file, which are less than or equal to 255 characters.

Remember that most EMACS commands can be given a repeat count. Note that this includes characters which insert themselves.

>> Try that now -- type ESC-8 * and see what happens.

If you want to insert spaces in a line, type ^C.

>> Move to a line and move the cursor with ^F's; then insert spaces with ^C. Use ^D to remove the spaces.

If you want to create a blank line between two lines, move to the second of the two lines and type ^O.

>> Try moving to a line and typing ^O now.

You've now learned the most basic way of typing something in EMACS and correcting errors. You can delete characters, words or lines as well.Here is a summary of the delete operations:

<Delete>	Delete the character just before the cursor
^H	Delete the character just before the cursor
^D	Delete the character the cursor is under
ESC-<Delete>	Kill the word immediately before the cursor
ESC-^H	Kill the word immediately before the cursor
ESC-D	Kill the word from the cursor position
^K	Kill from the cursor position to end of line

Notice that <Delete> and ^D vs ESC-<Delete> and ESC-D extend the parallel started by ^F and ESC-F (well, <Delete> isn't really a control character,but let's not worry about that).

Now suppose you kill something, and then you decide that you want to get it back? Well, whenever you kill something bigger than a character, EMACS saves it for you. To yank it back, use ^Y. Note that you don't have to be in the same place to do ^Y. This is a good way to move text around.Also note the difference between "Killing" and "Deleting" - "Killed" text can be yanked back, and "Deleted" text cannot. Generally, the commands that can destroy a lot of text save it, while the ones that attack only one character do not save it.

>> Type ^N a couple times to position the cursor at some line on this screen. Now kill that line with ^K.

Note that a single ^K kills the contents of the line, and a second ^K kills the line itself, and makes all the other lines move up. If you give ^K a repeat count, it kills that many lines AND their contents.

The text that has just disappeared is saved so that you can retrieve it. To retrieve the last killed text and put it where the cursor currently is, type ^Y.

>> Try it. Type ^Y to yank the text back.

Think of ^Y as if you were yanking something back that someone took away from you. Notice that if you do several ^K's in a row the text that is killed is all saved together so that one ^Y will yank all of the lines.

>> Try it. Type ^K several times.

>> To retrieve that killed text: Type ^Y. Move the cursor down a few lines and type ^Y again. You now know how to copy text.

What do you do if you have some text you want to yank back, and then you kill something else? ^Y would yank the more recent kill.

>> Kill a line, move around, kill another line. Then do ^Y to get back the second killed line.

SEARCHING

EMACS can do searches for strings (these are groups of contiguous characters or words) either forward through the file or backward through it.

>> Now type ^S to start a search. Type the word "cursor", then ESC.

>> Type ^S ESC to find the next occurrence of "cursor".

The ^S starts a search that looks for any occurrence of the search string AFTER the current cursor position. But what if you want to search for something earlier in the text? To do this one should type ^R for Reverse search. Everything that applies to ^S applies to ^R except that the direction of the search is reversed.

TEXT REPLACEMENT

>> Move the cursor to the blank line two lines below this one. Then type ESC-R changed ESC altered ESC . Notice how this line has changed; you have replaced the word "changed" with "altered" wherever it occurs in the file after the cursor. After all the substitutions have been made or the end of file has been reached, a message informing you of the number of substitutions which have been made appears in the communication line.

The more customary command for replacing strings is the interactive command query-replace-search (ESC-^R), which has several options. In essence, it shows each occurrence of the first string and asks you if you want to replace it or not. Type a "?" when it asks to replace the string to list the various options for query-replace-search. For a more detailed discussion of this command refer to the EMACS manual.

FILES

In order to make the text changes permanent, you must save them to a file.If you do not save them, the changes will "disappear" when you leave EMACS.As you make changes, i.e., corrections, deletions, insertions, etc., they are actually written to a "scratch" copy of the file and the changes to this

file will not affect the "master" copy of the file until a file save is specified. This allows you to decide ifchanges made to the file should be made permanent or discarded.

Remember: The file name appears on the mode line.

=*== MicroEMACS 3.9i () == emacs.tut == File: emacs.tut===

The commands for finding and saving files are unlike the other commands you have learned so far in that they consist of two characters - a ^X followed by another character which specifies the file command to be executed.

To find a file, type ^X^F. EMACS will then prompt you from the communication line for the name of the file. In response to the prompt,type the file name followed by a <Return> to indicate the file name has been entered. This command will tell EMACS to go find this file and load it. Its contents will then be displayed on the screen and you will be able to edit the file's contents.

To save any changes made to the file, type ^X^S. This tells EMACS to create a new version of the file which includes the changes you have made. When the save is complete, the number of lines saved will be displayed in the communication line.

If you edit a file and at some point decide to quit (i.e., ^X^C) without saving the changes, EMACS will remind you that changes have been made to the file and ask you if you really want to quit. Enter "N" to return to EMACS or "Y" to exit EMACS without saving the changes.

To create a file, just edit it "as if" it already existed. Then start typing in the text. When you ask to "save" the file, EMACS will really create the file with the text that you have entered. From then on, you can consider yourself to be editing an existing file.

It is not easy for you to test editing a file and continue with the tutorial. But you can always come back into the tutorial by starting it over and skipping forward. So, when you feel ready, you should try editing a file named "FOO", putting some text in it, and saving it; then exit EMACS and look at the file to be sure that it worked.

EXTENDING THE COMMAND SET

There are many, many more EMACS commands than could possibly be put on all the CONTROL and META characters. EMACS gets around this with the X (eXtend)command. There are two forms of this command:

^X	Character eXtend. Followed by one character.
ESC-X	Named command eXtend. Followed by a long name.

These are commands that are generally useful but used less than the commands you have already learned about. You have already seen two of them: the file commands ^X^F to Find and ^X^S to Save. Another example is the command to tell EMACS that you'd like to stop editing. The command to do this is ^X^C.

There are many ^X commands. Right now, the most helpful ones will be

^X^F	Find file.
^X^S	Save file.
^X^C	Quit EMACS.

This does not save your files automatically; however, if your files have been modified, EMACS asks if you really want to quit. The standard way to save and exit is ^X^S ^X^C.

Named eXtended commands are commands which are used even less frequently,or commands which are used only in certain modes. These commands areusually called "functions". An example is the function "apropos", which prompts for a keyword and then gives the names of all the functions that are apropos for that keyword. When you type ESC-X, EMACS prompts you from the communication line with ":" and you should type the name of the function you wish to call; in this case, "apropos". Just type "apr<Space>"and EMACS will complete the name. EMACS will ask you for a keyword or phrase and you type the string that you want information on.

>> Type ESC-X, followed by "apropos<Return>" or "apr<Space>". Then type "file" followed by a <Return>. Note: ESC-A is equivalent to the ECS-X "apropos" command.

>> To remove the "window" that was added, type ^X0 (zero).

FUNCTION KEYS

By now, you should be familiar with the format and meaning of some of the more common CONTROL and META commands. Because several of these commands are used frequently, they have been bound to the function keys, which are usually located on the left-hand side of the keyboard and labelled F1..F10. By pressing the appropriate function key, one can replace several keystrokes with a single keystroke, thus saving you time as you become familiar with their use.

The highlighted portion at the top of the screen lists the commands which are associated with each function key. Each function key supports two commands specified by fn or Fn where n = 1, 2,...10. The default commands are represented by fn and are defined on the left side of the screen; these commands are executed by pressing the appropriate function key. The secondary commands are represented by Fn and are defined on the right side of the screen; these commands are executed by pressing the <Shift> key and the appropriate function key at the same time.

>> Press f1 would ESC. Note the position of the cursor - "would" was located just as if ^S would ESC had been entered. Enter ^S would ESC to see for yourself.

>> Press F1 (<Shift> f1). Note the different appearance of the screen. You have toggled the function key list, i.e., "turned it off". To "turn it on", press F1 again.

>> Try using some of the other function keys to become familiar with their use. NOTE: Do NOT use f9 with this file as it would save any changes you may have made while using the tutorial.

GETTING MORE HELP

In this tutorial we have tried to supply just enough information to get you started using EMACS. There is so much available in EMACS that it would be impossible to explain it all here. However, you may want to learn more about EMACS since it has numerous desirable features that you don't know about yet.

The most basic HELP feature is the describe-key function which is available by typing ^X? and then a command character. EMACS prints one line in the communication line to tell what function is bound to that key.

>> Type ^X?^P. The message in the communication line should be something like "^P is bound to previous-line".

NOTE: Multi-character commands such as ^X^Z and ESC-V are also allowed after ^X? .

The describe-command function does not work - December 1986
Skip to the next section

The describe-command function (ESC-?) will prompt for the name of a function and print out the section from the manual about that command.When you are finished reading it, type a space or a ^G (quit) to bring your text back on the screen.

Now let's get more information about the previous-line command.

>> Type ESC-?^P. When you are finished reading the output, type <Space>.

The "name of the function" is important for people who are customizing EMACS. It is what appears in the EMACS CHART as the documentation for the command character.

CONCLUSION

Remember: To EXIT use ^X^C.

This tutorial is meant to be understandable to all new users, so if you found something unclear, don't sit and blame

yourself - complain!

You'll probably find that if you use EMACS for a few days you won't be able to give it up. Initially it may give you trouble. But remember,this is the case with any editor, especially one that can do many, many things - and EMACS can do practically everything.

ACKNOWLEDGEMENTS

This is a modified version of the "JOVE Tutorial" by Jonathan Payne(19 January 86). That document was in turn a modified version of the tutorial "Teach-Emacs" from MIT as modified by Steve Zimmerman at CCA-UNIX (31 October 85).

Update - February 1986 by Dana Hoggatt.

Update - December 1986 by Kim Leburg.

Update - November 1987, February 1988, January 1989 by Daniel Lawrence