Emacs Tutorial

NOTE: It is suggested that you start up Emacs before continuing. To bring up this tutorial in Emacs, hit the ESC key following by "x" then type help-with-tutorial followed by hitting the RETURN key.

Copyright (c) 1985 Free Software Foundation, Inc; See end for conditions.

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

C-<chr> means hold the CONTROL key while typing the character <chr> Thus, C-f would be: hold the CONTROL key and type f.

M-<chr> means hold the META or EDIT key down while typing <chr>. If there is no META or EDIT key, type <ESC>, release it, then type the character <chr>. "<ESC>" stands for the key labelled "ALT" or "ESC".

Important note: to end the Emacs session, type C-x C-c. (Two characters.)

The characters ">>" at the left margin indicate directions for you to try using a command. For instance:

>> Now type C-v (View next screen) to move to the next screen. (go ahead, do it by depressing the control key and v together). From now on, you'll be expected to do this whenever you finish reading the 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 C-v. To move backwards a screen, type M-v (depress the META key and type v, or type
v if you don't have a META or EDIT key). >> Try typing M-v and then C-v to move back and forth a few times.

Summary

The following commands are useful for viewing screenfuls:

```
        C-v     Move forward one screenful
        M-v     Move backward one screenful
        C-l     Clear screen and redisplay everything
                 putting the text near the cursor at the center.
                 (That's control-L, not control-1.)
```

>> Find the cursor and remember what text is near it. Then type a C-l. Find the cursor again and see what text is near it now.

Basic Cursor Control

Getting from screenful to screenful 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 C-p, C-b, C-f, and C-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, C-p
                         :
```

```
                                  :
      Backward, C-b .... Current cursor position .... Forward, C-f
                                  :
                                  :
                           Next line, C-n
```

>> Move the cursor to the line in the middle of that diagram and type C-l to see the whole diagram centered in the screen.

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 C-n's to bring the cursor down to this line.

>> Move into the line with C-f's and then up with C-p's. See what C-p does when the cursor is in the middle of the line.

Lines are separated by Newline characters. For most applications there should normally be a Newline character at the end of the text, as well, but it is up to you to make sure of this. A file can validly exist without a Newline at the end.

>> Try to C-b at the beginning of a line. Do a few more C-b's. Then do C-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.

>> Try to move the cursor off the bottom of the screen with C-n and see what happens.

If moving by characters is too slow, you can move by words. M-f (Meta-f) moves forward a word and M-b moves back a word.

>> Type a few M-f's and M-b's. Intersperse them with C-f's and C-b's.

Notice the parallel between C-f and C-b on the one hand, and M-f and M-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). There is a similar parallel between lines and sentences: C-a and C-e move to the beginning or end of a line, and M-a and M-e move to the beginning or end of a sentence.

>> Try a couple of C-a's, and then a couple of C-e's.
Try a couple of M-a's, and then a couple of M-e's.

See how repeated C-a's do nothing, but repeated M-a's keep moving farther. Do you think that this is right?

Two other simple cursor motion commands are M-< (Meta Less-than), which moves to the beginning of the file, and M-> (Meta Greater-than), which moves to the end of the file. You probably don't need to try them, since finding this spot again will be boring. On most terminals the "<" is above the comma and you must use the shift key to type it. On these terminals you must use the shift key to type M-< also; without the shift key, you would be typing M-comma. The location of the cursor in 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 sentence moving commands:

```
      C-f      Move forward a character
      C-b      Move backward a character

      M-f      Move forward a word
      M-b      Move backward a word
```

```
        C-n     Move to next line
        C-p     Move to previous line

        C-a     Move to beginning of line
        C-e     Move to end of line

        M-a     Move back to beginning of sentence
        M-e     Move forward to end of sentence

        M-<     Go to beginning of file
        M->     Go to end of file
```

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 typing C-u and then the digits before you type the command. If you have a META or EDIT key, you can omit the C-u if you hold down the META or EDIT key while you type the digits. This is easier, but we recommend the C-u method because it works on any terminal.

For instance, C-u 8 C-f moves forward eight characters.

>> Try giving a suitable argument to C-n or C-p to come as close as you can to this line in one jump.

The only apparent exception to this is the screen moving commands, C-v and M-v. When given an argument, they scroll the screen up or down by that many lines, rather than screenfuls. This proves to be much more useful.

>> Try typing C-u 8 C-v now.

Did it scroll the screen up by 8 lines? If you would like to scroll it down you can give an argument to M-v.

If you are using X Windows, there is probably a rectangular area called a scroll bar at the right hand side of the Emacs window. You can scroll the text by clicking the mouse in the scroll bar.

>> Try pressing the middle button at the top of the highlighted area within the scroll bar, then moving the mouse while holding that button down.

>> Move the mouse to a point in the scroll bar about three lines from the top, and click the left button a couple of times. Then try the right button a couple of times.

When Emacs is hung

If Emacs gets into an infinite (or simply very long) computation which you don't want to finish, you can stop it safely by typing C-g. You can also use C-g to discard a numeric argument or the beginning of a command that you don't want to finish.

>> Type C-u 100 to make a numeric arg of 100, then type C-g. Now type C-f. How many characters does it move? If you have typed an <ESC> by mistake, you can get rid of it with a C-g.

If you type <ESC> <ESC>, you get a new window appearing on the screen, telling you that M-ESC is a "disabled command" and asking whether you really want to execute it. The command M-ESC is marked as disabled because you probably don't want to use it until you know more about Emacs, and we expect it would confuse you if it were allowed to go ahead and run. If you really want to try the M-ESC command, you could type a Space in answer to the question and M-ESC would go ahead. Normally, if you do not want to execute M-ESC, you would type "n" to answer the question.

>> Type <ESC> <ESC>, then type n.

Windows

Emacs can have several windows, each displaying its own text. At this stage it is better not to go into the techniques of using multiple windows. But you do need to know how to get rid of extra windows that may appear to display help or output from certain commands. It is simple:

C-x 1 One window (i.e., kill all other windows).

That is Control-x followed by the digit 1. C-x 1 makes the window which the cursor is in become the full screen, by getting rid of any other windows.

>> Move the cursor to this line and type C-u 0 C-l.
>> Type Control-h k Control-f. See how this window shrinks, while a new one appears to display documentation on the Control-f command.

>> Type C-x 1 and see the documentation listing window disappear.

Inserting and Deleting

If you want to insert text, just type it. Characters which you can see, such as A, 7, *, etc. are taken by Emacs as text and inserted immediately. Type <Return> (the carriage-return key) to insert a Newline character.

You can delete the last character you typed by typing <Rubout>. <Rubout> is a key on the keyboard, which might be labelled "Delete" instead of "Rubout" on some terminals. More generally, <Rubout> deletes the character immediately before the current cursor position.

>> Do this now, type a few characters and then delete them by typing <Rubout> a few times. Don't worry about this file being changed; you won't affect the master tutorial. This is just a copy of it.

>> Now start typing text until you reach the right margin, and keep typing. When a line of text gets too big for one line on the screen, the line of text is "continued" onto a second screen line. The backslash at the right margin indicates a line which has been continued.

>> Use <Rubout>s to delete the text until the line fits on one screen line again. The continuation line goes away.

>> Move the cursor to the beginning of a line and type <Rubout>. This deletes the newline before the line and merges the line onto the previous line. The resulting line may be too long to fit, in which case it has a continuation line.

>> Type <Return> to reinsert the Newline you deleted.

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

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

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

```
        <Rubout>     delete the character just before the cursor
        C-d          delete the next character after the cursor

        M-<Rubout>   kill the word immediately before the cursor
        M-d          kill the next word after the cursor
```

```
        C-k             kill from the cursor position to end of line
        M-k             kill to the end of the current sentence
```

Notice that <Rubout> and C-d vs M-<Rubout> and M-d extend the parallel started by C-f and M-f (well, <Rubout> isn't really a control character, but let's not worry about that). C-k and M-k are like C-e and M-e, sort of, in that lines are opposite sentences.

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 C-y. You can kill text in one place, move elsewhere, and then do C-y; this is a good way to move text around. Note that the difference between "Killing" and "Deleting" something is that "Killed" things can be yanked back, and "Deleted" things cannot. Generally, the commands that can destroy a lot of text save it, while the ones that attack only one character, or nothing but blank lines and spaces, do not save.

For instance, type C-n a couple times to position the cursor at some line on this screen.

>> Do this now, move the cursor and kill that line with C-k.

Note that a single C-k kills the contents of the line, and a second C-k kills the line itself, and make all the other lines move up. If you give C-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 C-y.

>> Try it; type C-y to yank the text back.

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

>> Do this now, type C-k several times.

Now to retrieve that killed text:

>> Type C-y. Then move the cursor down a few lines and type C-y again. You now see how to copy some text.

What do you do if you have some text you want to yank back, and then you kill something else? C-y would yank the more recent kill. But the previous text is not lost. You can get back to it using the M-y command. After you have done C-y to get the most recent kill, typing M-Y replaces that yanked text with the previous kill. Typing M-y again and again brings in earlier and earlier kills. When you have reached the text you are looking for, you can just go away and leave it there. If you M-y enough times, you come back to the starting point (the most recent kill).

>> Kill a line, move around, kill another line. Then do C-y to get back the second killed line. Then do M-y and it will be replaced by the first killed line. Do more M-y's and see what you get. Keep doing them until the second kill line comes back, and then a few more. If you like, you can try giving M-y positive and negative arguments.

Undo

Any time you make a change to the text and wish you had not done so, you can undo the change (return the text to its previous state) with the undo command, C-x u. Normally, C-x u undoes one command's worth of changes; if you repeat the C-x u several times in a row, each time undoes one more command. There are two exceptions: commands that made no change (just moved the cursor) do not count, and self-inserting characters are often lumped together in groups of up to 20. This is to reduce the number of C-x u's you have to type.

C-_ is another command for undoing; it is just the same as C-x u but easier to type several times in a row. The problem with C-_ is that on some keyboards it is not obvious how to type it. That is why C-x u is provided as well. On some DEC terminals, you can type C-_ by typing / while holding down CTRL. Illogical, but what can you expect from DEC?

Giving a numeric argument to C-_ or C-x u is equivalent to repeating it as many times as the argument says.

Files

In order to make the text you edit permanent, you must put it in a file. Otherwise, it will go away when your invocation of Emacs goes away. You put your editing in a file by "finding" the file. What finding means is that you see the contents of the file in your Emacs; and, loosely speaking, what you are editing is the file itself. However, the changes still don't become permanent until you "save" the file. This is so you can have control to avoid leaving a half-changed file around when you don't want to. Even then, Emacs leaves the original file under a changed name in case your changes turn out to be a mistake.

If you look near the bottom of the screen you will see a line that begins and ends with dashes, and contains the string "Emacs: TUTORIAL". Your copy of the Emacs tutorial is called "TUTORIAL". Whatever file you find, that file's name will appear in that precise spot.

The commands for finding and saving files are unlike the other commands you have learned in that they consist of two characters. They both start with the character Control-x. There is a whole series of commands that start with Control-x; many of them have to do with files, buffers, and related things, and all of them consist of Control-x followed by some other character.

Another thing about the command for finding a file is that you have to say what file name you want. We say the command "reads an argument from the terminal" (in this case, the argument is the name of the file). After you type the command

C-x C-f Find a file

Emacs asks you to type the file name. It echoes on the bottom line of the screen. You are using the minibuffer now! this is what the minibuffer is for. When you type <Return> to end the file name, the minibuffer is no longer needed, so it disappears.

>> Type C-x C-f, then type C-g. This cancels the minibuffer, and also cancels the C-x C-f command that was using the minibuffer. So you do not find any file.

In a little while the file contents appear on the screen. You can edit the contents. When you wish to make the changes permanent, issue the command

C-x C-s Save the file

The contents of Emacs are written into the file. The first time you do this, the original file is renamed to a new name so that it is not lost. The new name is made by appending "~" to the end of the original file's name.

When saving is finished, Emacs prints the name of the file written. You should save fairly often, so that you will not lose very much work if the system should crash.

>> Type C-x C-s, saving your copy of the tutorial. This should print "Wrote .../TUTORIAL" at the bottom of the screen. On VMS it will print "Wrote ...[...]TUTORIAL."

To make a new file, just find 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 inserted. From then on, you can

consider yourself to be editing an already existing file.

Buffers

If you find a second file with C-x C-f, the first file remains inside Emacs. You can switch back to it by finding it again with C-x C-f. This way you can get quite a number of files inside Emacs.

The object inside Emacs which holds the text read from one file is called a "buffer." Finding a file makes a new buffer inside Emacs. To see a list of the buffers that exist in Emacs, type

C-x C-b List buffers

>> Try C-x C-b now.

See how each buffer has a name, and it may also have a file name for the file whose contents it holds. Some buffers do not correspond to files. For example, the buffer named "*Buffer List*" does not have any file. It is the buffer which contains the buffer list that was made by C-x C-b. ANY text you see in an Emacs window has to be in some buffer.

>> Type C-x 1 to get rid of the buffer list.

If you make changes to the text of one file, then find another file, this does not save the first file. Its changes remain inside Emacs, in that file's buffer. The creation or editing of the second file's buffer has no effect on the first file's buffer. This is very useful, but it also means that you need a convenient way to save the first file's buffer. It would be a nuisance to have to switch back to it with C-x C-f in order to save it with C-x C-s. So we have

C-x s Save some buffers

C-x s goes through the list of all the buffers you have and finds the ones that contain files you have changed. For each such buffer, C-x s asks you whether to save it.

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. This comes in two flavors:

```
        C-x      Character eXtend.  Followed by one character.
        M-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 C-x C-f to Find and C-x C-s to Save. Another example is the command to tell Emacs that you'd like to stop editing and get rid of Emacs. The command to do this is C-x C-c. (Don't worry; it offers to save each changed file before it kills the Emacs.)

C-z is the usual way to exit Emacs, because it is always better not to kill the Emacs if you are going to do any more editing. On systems which allow it, C-z exits from Emacs to the shell but does not destroy the Emacs; if you use the C shell, you can resume Emacs with the `fg' command (or, more generally, with `%emacs', which works even if your most recent job was some other). On systems where suspending is not possible, C-z creates a subshell running under Emacs to give you the chance to run other programs and return to Emacs afterward, but it does not truly "exit" from Emacs. In this case, the shell command `exit' is the usual way to get back to Emacs from the subshell.

You would use C-x C-c if you were about to log out. You would also use it to exit an Emacs invoked under mail handling programs and other random utilities, since they may not believe you have really

finished using the Emacs if it continues to exist.

There are many C-x commands. The ones you know are:

```
        C-x C-f         Find file.
        C-x C-s         Save file.
        C-x C-b         List buffers.
        C-x C-c         Quit Emacs.
        C-x u           Undo.
```

Named eXtended commands are commands which are used even less frequently, or commands which are used only in certain modes. These commands are usually called "functions". An example is the function replace-string, which globally replaces one string with another. When you type M-x, Emacs prompts you at the bottom of the screen with M-x and you should type the name of the function you wish to call; in this case, "replace-string". Just type "repl s<TAB>" and Emacs will complete the name. End the command name with <Return>. Then type the two "arguments"--the string to be replaced, and the string to replace it with--each one ended with a Return.

>> Move the cursor to the blank line two lines below this one. Then type M-x repl s<Return>changed<Return>altered<Return>.

Notice how this line has changed: you've replaced the word c-h-a-n-g-e-d with "altered" wherever it occurred after the cursor.

Mode Line

If Emacs sees that you are typing commands slowly it shows them to you at the bottom of the screen in an area called the "echo area." The echo area contains the bottom line of the screen. The line immediately above it is called the MODE LINE. The mode line says something like

```
--**-Emacs: TUTORIAL             (Fundamental)--58%----------------------
```

This is a very useful "information" line.

You already know what the filename means--it is the file you have found. What the --NN%-- means is that NN percent of the file is above the top of the screen. If the top of the file is on the screen, it will say --Top-- instead of --00%--. If the bottom of the file is on the screen, it will say --Bot--. If you are looking at a file so small it all fits on the screen, it says --All--.

The stars near the front mean that you have made changes to the text. Right after you visit or save a file, there are no stars, just dashes.

The part of the mode line inside the parentheses is to tell you what modes you are in. The default mode is Fundamental which is what you are in now. It is an example of a "major mode". There are several major modes in Emacs for editing different languages and text, such as Lisp mode, Text mode, etc. At any time one and only one major mode is active, and its name can always be found in the mode line just where "Fundamental" is now. Each major mode makes a few commands behave differently. For example, there are commands for creating comments in a program, and since each programming language has a different idea of what a comment should look like, each major mode has to insert comments differently. Each major mode is the name of an extended command, which is how you get into the mode. For example, M-x fundamental-mode is how to get into Fundamental mode.

If you are going to be editing English text, such as this file, you should probably use Text Mode.

>> Type M-x text-mode<Return>.

Don't worry, none of the commands you have learned changes Emacs in any great way. But you can observe that apostrophes are now part of words when you do M-f or M-b. Major modes are usually like

that: commands don't change into completely unrelated things, but they work a little bit differently.

To get documentation on your current major mode, type C-h m.

>> Use C-u C-v once or more to bring this line near the top of screen.
>> Type C-h m, to see how Text mode differs from Fundamental mode.
>> Type C-x 1 to remove the documentation from the screen.

Major modes are called major because there are also minor modes. They are called minor because they aren't alternatives to the major modes, just minor modifications of them. Each minor mode can be turned on or off by itself, regardless of what major mode you are in, and regardless of the other minor modes. So you can use no minor modes, or one minor mode, or any combination of several minor modes.

One minor mode which is very useful, especially for editing English text, is Auto Fill mode. When this mode is on, Emacs breaks the line in between words automatically whenever the line gets too long. You can turn this mode on by doing M-x auto-fill-mode<Return>. When the mode is on, you can turn it off by doing M-x auto-fill-mode<Return>. If the mode is off, this function turns it on, and if the mode is on, this function turns it off. This is called "toggling".

>> Type M-x auto-fill-mode<Return> now. Then insert a line of "asdf " over again until you see it divide into two lines. You must put in spaces between them because Auto Fill breaks lines only at spaces.

The margin is usually set at 70 characters, but you can change it with the C-x f command. You should give the margin setting you want as a numeric argument.

>> Type C-x f with an argument of 20. (C-u 2 0 C-x f). Then type in some text and see Emacs fill lines of 20 characters with it. Then set the margin back to 70 using C-x f again.

If you makes changes in the middle of a paragraph, Auto Fill mode does not re-fill it for you. To re-fill the paragraph, type M-q (Meta-q) with the cursor inside that paragraph.

>> Move the cursor into the previous paragraph and type M-q.

Searching

Emacs can do searches for strings (these are groups of contiguous characters or words) either forward through the file or backward through it. To search for the string means that you are trying to locate it somewhere in the file and have Emacs show you where the occurrences of the string exist. This type of search is somewhat different from what you may be familiar with. It is a search that is performed as you type in the thing to search for. The command to initiate a search is C-s for forward search, and C-r for reverse search. BUT WAIT! Don't do them now. When you type C-s you'll notice that the string "I-search" appears as a prompt in the echo area. This tells you that Emacs is in what is called an incremental search waiting for you to type the thing that you want to search for. <RET> terminates a search.

>> Now type C-s to start a search. SLOWLY, one letter at a time, type the word 'cursor', pausing after you type each character to notice what happens to the cursor.
>> Type C-s to find the next occurrence of "cursor".
>> Now type <Rubout> four times and see how the cursor moves.
>> Type <RET> to terminate the search.

Did you see what happened? Emacs, in an incremental search, tries to go to the occurrence of the string that you've typed out so far. To go to the next occurrence of 'cursor' just type C-s again. If no such occurrence exists Emacs beeps and tells you that it is a failing search. C-g would also terminate the search.

If you are in the middle of an incremental search and type <Rubout>, you'll notice that the last character in the search string is erased and the search backs up to the last place of the search. For instance, suppose you

currently have typed 'cu' and you see that your cursor is at the first occurrence of 'cu'. If you now type <Rubout>, the 'u' on the search line is erased and you'll be repositioned in the text to the occurrence of 'c' where the search took you before you typed the 'u'. This provides a useful means for backing up while you are searching.

If you are in the middle of a search and type a control or meta character (with a few exceptions--characters that are special in a search, such as C-s and C-r), the search is terminated.

The C-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, type C-r for Reverse search. Everything that applies to C-s applies to C-r except that the direction of the search is reversed.

Multiple Windows

One of the nice features of Emacs is that you can display more than one window on the screen at the same time.

>> Move the cursor to this line and type C-u 0 C-l.

>> Now type C-x 2 which splits the screen into two windows. Both windows display this tutorial. The cursor stays in the top window.

>> Type C-M-v to scroll the bottom window.

>> Type C-x o ("o" for "other") to move the cursor to the bottom window.

>> Use C-v and M-v in the bottom window to scroll it. Keep reading these directions in the top window.

>> Type C-x o again to move the cursor back to the top window. The cursor is still just where it was in the top window before.

You can keep using C-x o to switch between the windows. Each window has its own cursor position, but only one window actually shows the cursor. All the ordinary editing commands apply to the window that the cursor is in.

The command C-M-v is very useful when you are editing text in one window and using the other window just for reference. You can keep the cursor always in the window where you are editing, and edit there as you advance through the other window.

>> Type C-x 1 (in the top window) to get rid of the bottom window.

(If you had typed C-x 1 in the bottom window, that would get rid of the top one. Think of this command as "Keep just one window--the window I am already in.")

You don't have to display the same buffer in both windows. If you use C-x C-f to find a file in one window, the other window doesn't change. You can pick a file in each window independently.

Here is another way to use two windows to display two different things:

>> Type C-x 4 C-f followed by the name of one of your files. End with <RETURN>. See the specified file appear in the bottom window. The cursor goes there, too.

>> Type C-x o to go back to the top window, and C-x 1 to delete the bottom window.

Recursive Editing Levels

Sometimes you will get into what is called a "recursive editing level". This is indicated by square brackets in the mode line, surrounding the parentheses around the major mode name. For example, you might see [(Fundamental)] instead of (Fundamental).

To get out of the recursive editing level, type

M-x top-level<Return>.

>> Try that now; it should display "Back to top level" at the bottom of the screen.

In fact, you were ALREADY at top level (not inside a recursive editing level) if you have obeyed instructions. M-x top-level does not care; it gets out of any number of recursive editing levels, perhaps zero, to get back to top level.

You can't use C-g to get out of a recursive editing level because C-g is used for discarding numeric arguments and partially typed commands WITHIN the recursive editing level.

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. Emacs has a great deal of internal documentation. All of these commands can be accessed through the character Control-h, which we call "the Help character" because of the function it serves.

To use the HELP features, type the C-h character, and then a character saying what kind of help you want. If you are REALLY lost, type C-h ? and Emacs will tell you what kinds of help it can give. If you have typed C-h and decide you don't want any help, just type C-g to cancel it.

The most basic HELP feature is C-h c. Type C-h, a c, and a command character or sequence, and Emacs displays a very brief description of the command.

>> Type C-h c Control-p. The message should be something like C-p runs the command previous-line

This tells you the "name of the function". That is important in writing Lisp code to extend Emacs; it also is enough to remind you of what the command does if you have seen it before but did not remember.

Multi-character commands such as C-x C-s and (if you have no META or EDIT key) <ESC>v are also allowed after C-h c.

To get more information on the command, use C-h k instead of C-h c.

>> Type C-h k Control-p.

This displays the documentation of the function, as well as its name, in an Emacs window. When you are finished reading the output, type C-x 1 to get rid of the help text. You do not have to do this right away. You can do some editing while referring to the help text and then type C-x 1.

Here are some other useful C-h options:

```
    C-h f        Describe a function.  You type in the name of the
                 function.


    C-h a        Command Apropos.  Type in a keyword and Emacs will list
                 all the commands whose names contain that keyword.
                 These commands can all be invoked with Meta-x.
```

```
                    For some commands, Command Apropos will also list a one
                    or two character sequence which has the same effect.
```

>> Type C-h a file<Return>.

This displays in another window a list of all M-x commands with "file" in their names. You will also see commands like C-x C-f and C-x C-w, listed beside the command names find-file and write-file.

Conclusion

Remember, to exit Emacs permanently use C-x C-c. To exit to a shell temporarily, so that you can come back in, use C-z.

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

Copying

This tutorial descends from a long line of Emacs tutorials starting with the one written by Stuart Cracraft for the original Emacs.

This version of the tutorial, like GNU Emacs, is copyrighted, and comes with permission to distribute copies on certain conditions:

Copyright (c) 1985 Free Software Foundation

Permission is granted to anyone to make or distribute verbatim copies of this document as received, in any medium, provided that the copyright notice and permission notice are preserved, and that the distributor grants the recipient permission for further redistribution as permitted by this notice.

Permission is granted to distribute modified versions of this document, or of portions of it, under the above conditions, provided also that they carry prominent notices stating who last altered them.

The conditions for copying Emacs itself are slightly different but in the same spirit. Please read the file COPYING and then do give copies of GNU Emacs to your friends. Help stamp out software obstructionism ("ownership") by using, writing, and sharing free software!

Press here to return to the Editors Menu