THE SURVIVAL GUIDE TO Emacs
version 1.0

Lluís A. Belanche
Dept. de Llenguatges i Sistemes Informàtics.
Universitat Politècnica de Catalunya.
belanche@lsi.upc.es

September 8, 2003

Little disclaimer: this document is by no means intended to be a complete or detailed guide through Emacs nor to replace the built-in help. Rather, I wrote it as a modest introduction to the basic commands directed to those people new to the editor or wanting to possibly learn new useful commands. Feel free to distribute it. Send any comments or suggestions to the address above.

1 Introduction

If you came to reading this, you probably know Emacs is a text editor (though it can handle tar files too). To use it, type "Emacs <filename>". If a file named <filename> exists in your current directory, it will be opened; if not, a welcome message will be displayed.

In what follows, I use the convention of writing [*something*] meaning "type that *something* on the keyboard". I know this may sometimes be redundant, but it is a safe way to know what has to be typed exactly. There are two fundamental keys in Emacs : the CONTROL and the META keys, which allow to supply commands to the editor in a quick way, as follows:

- The CONTROL key is right in your keyboard. From now on, [C-] will refer to this key so that, for instance, [C-x] means "hold the CONTROL key while pressing x". Note [C-x u] means [C-x], releasing the CONTROL key and [u], so is not the same as [C-xu] (which means "hold the CONTROL key while typing [xu]" and is tantamount to [C-x C-u]).
- The META key can take several forms. In some keyboards, it refers to the EDIT key, but in most modern ones, it refers to the ESCAPE (or ESC) key or the ALT key. If you have both ESC and ALT keys, you can use them interchangeably as the META key. However, while ALT is used like CONTROL (holding it), ESC is pressed just as any other key.

All Emacs commands are obtainable by typing the "universal command executor" [M-x] followed by the command name (if you know it). If you don't, but know it partially, the TAB key will display a list of possible completions. In this document, however, we will concentrate on basic key bindings (sequences of keystrokes which perform these commands). A list of all key bindings can be obtained as [M-x describe-bindings]. For example, the key sequence [M-%] (described below) is a shorthand for [M-x query-replace].

CONTENTS:

2 Preliminaries

If you are using X Windows, the Emacs Window will use the sides to further interact with you. On the left hand side, there is a scroll bar, in which you can scroll the text by clicking the mouse in there. There should be a highlighted segment which roughly indicates the size of the text you are viewing (relative to the entire file) and its relative position on the file. Use the left button for moving screens of text, and the middle button for fast moving.

On top of the Emacs Window you will find a typical Menu chart (with options, suboptions, etc) to be spanned with the mouse. Many of them are accompanied by the corresponding keystrokes (the ones we will try to learn here). Emacs can be commanded with these Menus but many things can be done quicker and easier with the keystrokes. Try experimenting with the Menus a little if you wish. Also good to know is [C-z], which iconifies the entire Emacs Window.

The Emacs Window bottom lines are specially useful. There is one displaying a number of changing items, called the “mode line”. It tells you information about the status of Emacs and the text you are editing and looks something like this:

```
--:**  myfile.tex 13:28 0.04 Mail (LaTeX Ovwrt Fill)--L103--15%--------------
```

The two stars mean there are unsaved changes (otherwise –:– would be displayed), then comes the name of the file you are viewing (actually the buffer, see later), two optional items (time and incoming Mail), the editing mode (in this case Emacs knows it is a LaTeX file, overwrite is ON and auto-fill is also ON), and then two ‘position’ items. The L103 is telling you the line number (relative to the top of the file) in which the cursor is right now. The 15% is the percentage of the file above this line (compare with the above mentioned highlighted segment).

Underneath the “mode line” is the “echo area”. This line is used by Emacs to communicate some messages to the user, as well as to display slowly typed commands (in order for you to see them while typing them).

Emacs displays a predefined number n of characters per line on the screen, and then either the line is wrapped (if auto-fill-mode is on, which is indicated by the character ‘\’ at the end of

every wrapped line, and also at the Emacs Window bottom line) or a new line automatically begins (auto-fill-mode off). A new line begins in any case by hitting the RETURN key.
The involved basic commands are as follows:

[M-x auto-fill-mode] *// set auto-fill mode on, at about the 70 character mark (default n).*
// Executing the command again turns the mode off
[C-u <number> C-x f] *// change n to <number>*
[M-q] *// reformat current paragraph, according to current settings*
[M-g] *// reformat selected region (see below), according to current settings*

3 Text manipulation

3.1 Cursor movement

[C-p] *// move one character UP (from (p)revious)*
[C-n] *// move one character DOWN (from (n)ext)*
[C-b] *// move one character LEFT (from (b)ackward)*
[C-f] *// move one character RIGHT (from (f)orward)*

Why are these commands useful if one has the arrow keys? I find them useful when typing several "control-something" keys, so that I don't have to press and depress the control key. They can also be useful in defining commands for repetition (see below).

Note the UP and DOWN movements always lead you to the immediately following line, but at the end of the line if this was shorter than the one you come from (try it!). This is actually very convenient. Want to go faster? You can step entire words in the same way by using the META key instead:

[M-b] *// move one word LEFT the current (from (b)ackward)*
[M-f] *// move one word RIGHT the current (from (f)orward)*

However, and unlike those above, these commands do not auto-repeat, i.e., if you want to move three words ahead, you have to type the sequence [M-f] three times. There is a way to automatize this telling Emacs to process a sequence (a command) a number of times (described below), but in this case it is much easier to use the following:

[C-$\leftarrow$] *// move one word LEFT*
[C-$\rightarrow$] *// move one word RIGHT*

which do the autorepeat (where $\leftarrow$ and $\rightarrow$ are the left and right ARROW keys). You can go really fast this way (but of course if you want to move to a precise position that is far away your current position on the file, you'd better use the mouse).
A very useful set of commands lead you to the beginning/end of the current line/sentence:

[C-a] *// move to beginning of line*
[C-e] *// move to end of line*
[M-a] *// move to beginning of sentence*
[M-e] *// move to end of sentence*

A very convenient movement is a screenful, ideally suited for reading text. This is normally accomplished by using the NEXTPAGE and PREVPAGE keys. It can also be done in Emacs by just in case:

[C-v] // *(v)iew next screen*
[M-v] // *(v)iew previous screen*

You will notice there is an overlap of two lines when you move from screen to screen in order to provide some continuity.
Finally, to go to the beginning/end of the entire file, use:

[M-<] // *move to beginning of file*
[M->] // *move to end of file*

3.2 Deletion from cursor

We are used to delete *backwards* (simply using the delete (DEL) key) since this is the natural way, but computer editors allow to delete whatever is *forward* (ahead of the current position). Once you get used you will find that it sometimes allows you to make some corrections faster.

[C-d] // *erase one character forward (from (d)elete)*

whereas word deleting is accomplished again by using META instead of CONTROL:

[M-d] // *erase one word forward (from (d)elete)*
[M-DEL] // *erase one word backward (from (d)elete)*

Sometimes we wish to erase everything from the current position to the end of some text unit:

[M-d] // *erase to the end of word (from (d)elete)*
[C-k] // *erase to the end of line (from (k)ill)*
[M-k] // *erase to the end of sentence (from (k)ill)*

As you may have noticed, these last three commands can be used to erase *the whole* text unit (word, line, sentence). Just move to the beginning of the unit. That is why [M-d] erases one word forward. For example, to erase the whole current line, move to its beginning and erase to the end: [C-a] [C-k], which can be done quickly by keeping the control key pressed while typing [ak], which we write [C-ak]. Deletion of arbitrary portions of text involves the definition of a *region* and is covered in the next subsection.

A final speciality is char zapping, to delete any text from current position to the first forward occurrence of a given character:

[M-z <char>] // *erase to first occurrence of <char>*

3.3 Copying or moving text

For small sized portions of text (partial words or lines) I usually delete them (in order for Emacs to store them), then yank them in place (in case I wanted a copy), move to the desired new location and yank them again (of course this yank operation can be performed more than two times).

For more than small portions or portions without clear-cut limits (part of a sentence or paragraph, or an entire section of text), it is better to define a *region* and make Emacs store it. This region is defined by selecting the begin and end positions ("marks") in two ways:

1. with the keyboard, use [C-SPACE] for the first mark (Emacs types 'Mark set' at the Window bottom line) and move to the location of the desired second mark. The region is now defined.

2. with the mouse, simply go to the desired place for the first mark and hold the left button pressed while moving to the second mark, then release the button.

Alternatively, you can set the first mark with [C-SPACE] and the second by left-clicking with the mouse on the desired position. This is sometimes quicker but involves using the keyboard as well as the mouse, up to you! Anyway, once the region is defined, you can delete it with [C-w] and/or copy it with [C-y] or using the mouse middle button.

If you forgot where was the first mark when you are about to set the second mark, just use [C-xx]; this will set the second mark and position the cursor back on the first.

I strongly recommend to tell Emacs to highlight any region that is being defined, so that one can easily see if the region is actually what one wants. This is done by typing [M-x transient-mark-mode] (or alternatively using the 'Help/Options' menu on top of the window). You can select any region, from one character to the entire buffer. By the way, double left-clicking when in the middle of a word selects it. These are some useful handies:

[M-h] // *mark current paragraph*
[C-xp] // *mark current page*
[C-x h] // *mark entire buffer*

Sometimes it is necessary to select rectangular regions of text (*rectangles*), by which we mean portions of consecutive lines of equal length, but not necessarily spanning the whole line (which is the case in conventional regions). These rectangles have to be selected in the usual way, with two marks, and can then be routinely copied and/or moved to another position of the same or another buffer. Here are the basic commands for doing this:

[C-x r k] // *(k)ill (r)ectangle*
[C-x r y] // *(y)ank previously killed (r)ectangle*

The first mark is set as usual, with [C-SPACE]. This defines one of the corners of the rectangle. Then move the cursor to the opposite corner and do the kill. A rectangle can be "(c)leared" (filled with blanks) by doing [C-x r c] or its contents replaced with a specific string by doing [C-x r t] followed by the desired string.

A useful mechanism is that of *registers*, wherein you can store different regions (even rectangles) of text for later recovery. Just select the region normally and then proceed as follows:

[C-x r s <number>] // *(s)tore region (not rectangle) at (r)egister <number>*
[C-x r r <number>] // *store (r)ectangle at (r)egister <number>*
[C-x r i <number>] // *(i)nsert whatever text was previously stored at (r)egister <number>*

Registers can also be used to save positions in the buffer; this can be particularly useful when moving back and forth to specific places in a file:

[C-x r SPACE <number>] // *store current position at (r)egister <number>*
[C-x r j <number>] // *(j)ump to position previously stored at (r)egister <number>*

Up to 10 memories (numbered from 0 to 9) can be independently stored and recovered in the registers.

3.4 Text recovery

The effect of any deleting command (except DEL and [C-d]) can be recovered by typing [C-y] (from (y)ank), even if this was only a single character. If you now type [M-y] the immediate previously deleted item is inserted in the current position, *replacing* the one recovered by [C-y]. This can be repeated as long as there are items to be recovered, in a cyclic way. This way, you can recover text items deleted long time ago (in the same editing session) in other positions of the file.

3.5 Searching & replacing text

Emacs provides with a powerful mechanism for doing searches and replacements:

[C-s] // *incremental search forward*
[C-r] // *incremental search backward*

The searches are *incremental* in the sense that the search proceeds as you type consecutive characters of the text to search (it can be *any* subtext, not just single words). This means that, as you type, the search is more and more specific. Imagine you are in the middle of a search and you have just typed "hous": at this moment Emacs has left the cursor just at the first appearance of "hous" in the current buffer (forward or backward, but *from the position where the search started*); then you can do several things:

- type [C-s] or [C-r] and the closest appearance of "hous" (if any) will be located, in the specified direction. This can be done as many times as desired. Shall the end (or beginning) of the buffer be encountered, the search will proceed ciclycally (note Emacs warns with a 'Failing I-Search' message, just insist).
- type another character, say [e], then Emacs will proceed searching for "house" (in the last specified direction) and so forth.
- press the DEL key, then Emacs will undo the search of the last character. In our example, it will bring you back to the last found appearance of "hou" (i.e., just before typing the 's') and so forth.
- type [C-g], this will end the search and leave you back to the starting position. To simply end the search, press the RETURN key.

Note you can always change direction in the middle of the search. Also note that, even if the last search was performed some time ago, pressing [C-s] or [C-r] will resume it (from the current position). If you want to search for a previously searched text, use [M-p] (from (p)revious) right after [C-s] or [C-r]. Doing [M-p] several consecutive times will cycle through all searched texts. There is a basic but very useful command for text substitution:

[M-% <text1> RETURN <text2> RETURN]   // *replace <text1> by <text2>*

You will have to respond, at any moment a match is found, with one of the following:

[SPACE] or [y] to replace current match
[DELETE] or [n] to skip to next
[RETURN] or [q] to exit
[.] to replace current match and exit
[!] to replace all remaining matches with no more questions
[ˆ] to move back to previous match.

There is also a powerful replacement mechanism based on regular expressions:

[M-#] <regexp1> RETURN <regexp2> RETURN   // *replace regular expression <regexp1> by regular expression <regexp2>*

As a simple example, consider <regexp1> to be

```
\cite{\([.]+\)}
```

and <regexp2> to be

```
\citeauthor{\1}
```

Roughly, all the appearances of a string beginning with "`\cite{`", followed by a `group` of one or more single characters (signaled by [.]+ and numbered '1'), and ending in "}" are replaced by the string "`\citeauthor{group}`".

The command is obtainable as [M-x query-replace-regexp]. Regular expressions can also be used in incremental searches. The full syntax for regular expressions can be found in the accompanying Emacs Reference Card.

4 Command edition

4.1 Command undoing & aborting

The basic undo command is:

[C-x u]   // *undo last command*
To undo several consecutive commands, we have three options:

1. Use [C-x u] as many times as needed. This is a slow sequence to type, so it is recommended when careful undoing is desired.

2. Use instead the sequence [C-_] (this involves pressing the CONTROL and the SHIFT keys and hold them pressed while pressing the – key as many times as needed). This may be advantageous for quick undoing (it can be performed with the right hand using the right SHIFT key usually located next to the – key). Emacs will perform the undo's until a messsage "No more undo information" shall appear.

3. Use a repetition command (see below), like [C-u 5 C-x u].

How to undo the undo's? Simply press one of the ARROW keys and the undoing sequence will reverse. Why? when you press a key, the undoing process stops, so if you begin undoing normally again, the first undone steps will be precisely the undoing steps you just performed.

Any command sequence that you started to type can be aborted by typing [C-g] (Emacs responds by beeping and writing "Quit").

4.2 Command repetition

Use [C-u *n* <command>] where *n* is the number of times <command> has to be performed. Examples:

[C-u 82 C-n] *// move exactly 82 lines down*
[C-u 100 *] *// insert a row of 100 stars*
[C-u 3 META-y] *// insert the third more recently deleted item*

Assuming you have been using the ESC key for META, recall that the same effect can be obtained using the ALT key (which may sometimes be more convenient). For instance, typing a number while keeping the ALT key pressed is pretty quick, as in [ALT-56 *].

5 File management

Emacs maintains several directory files: the main one is the one you are currently editing (say "myfile.txt"). Every time you make changes to it and save them into "myfile.txt", Emacs creates a copy of the previous contents under the name "myfile.txt~". Further, it periodically writes an "auto save" file, saving all changes in "myfile.txt" that you have not saved yet, under the name "#myfile.txt#". This file is destroyed when you save the file in the normal way. Shall the computer crash in the middle of your work, you can later recover your auto-saved file by opening "myfile.txt" normally and then typing [M-x recover file].
These are the most common commands for file management within Emacs :

[C-xc] *// exit* Emacs *(asks for confirmation if any buffer not saved)*
[C-xs] *// (s)ave current buffer (does not exit* Emacs *)*
[C-x s] *// (s)ave all unsaved buffers (does not exit* Emacs *)*
[C-xw] *// (s)ave current buffer with ne(w) name (does not exit* Emacs *)*
[C-xv] *// load (read) an existing text file (replaces the one in the current buffer)*
[C-xf] *// same but reads into a di(f)fferent buffer*
[C-x i] *// (i)nserts an existing text file into the current buffer (from current position)*

6 Getting help from Emacs

[C-h i] *// displays* Emacs *help buffer*
[C-h m] *// help on current file mode*

In particular, after getting into the help buffer, [F] (note upcase) displays the Emacs FAQ (frequently asked questions) in a different buffer, whereas [t] displays (again in a different buffer) the Emacs Tutorial file (these correspond to [C-h F] and [C-h t], respectively).

What is a 'buffer'? Well, Emacs allows you to work with different portions of one or more text files simultaneously, although you can only *view* (and edit) one portion at a time. To do this, Emacs maintains several open buffers that you can switch to. Another possibility is to make more than one (usually two) buffers visible in the Emacs window (this is called split window). Other users may find more useful to invoke Emacs several times in different windows and use only one buffer for each (I personally prefer that). Anyway, these are the basic commands:

[C-x b] *// select or open another (b)uffer (default is previous)*
[C-xb] *// list all open (b)uffers*
[C-x k] *// (k)ill an open buffer*
[C-x 2] *// enter split window mode*
[C-x o] *// switch to (o)ther window (also with the mouse)*
[C-x 1] *// back to single window (keeps the buffer where the cursor was)*

If you ever get lost in the different Emacs buffers, you can always go back to the file you were editing using the 'Buffers' menu on top of the window.

7 Miscellaneous

This is a list of useful commands that do not deserve a subsection yet (maybe in the next version of the guide):

[C-t] *// (t)ranspose current character with previous*
[M-t] *// (t)ranspose current word (with previous if on the first character of a word, with next otherwise*
[C-xt] *// (t)ranspose current line with previous*
[M-!] *// execute a shell command*
[M-x shell] *// start a shell in new buffer*
[C-xl] *// convert a region to (l)owcase*
[C-xu] *// convert a region to (u)pcase*
[C-x l] *// (l)ines before current + (l)ines after current*
[C-l] *// redisplay screen, moving the text currently around the cursor to the center of the screen*

A special mention goes for the ".emacs" file. This is a text file that Emacs reads when invoked just before passing control to the user. It may be used to give some initial commands one wants to be executed every time. It may be that you open this file (using Emacs) and there is some text inthere; just add yours at the end. In particular, I have the following definitions:

```
(global-set-key "\eg" 'goto-line)
(global-set-key "\es" 'search-forward-regexp)
(global-set-key "\er" 'search-backward-regexp)
(global-set-key "\e#" 'query-replace-regexp)
```

This tells Emacs to define [M-g] as the "go to line" command, which asks for a line number (it also keeps a history that can be cycled with the arrow keys). The other correspond to the above-mentioned search and text replacement using regular expressions.