Emacs Howto Tutorial

According to a description at GNU.org, Emacs is the extensible, customizable, self-documenting real-time display editor. It offers true LISP -- smoothly integrated into the editor -- for writing extensions and provides an interface to the X Window System.

It has also been said (perhaps not entirely in jest) that Emacs can do so very many different things so well that it would make a fine operating system indeed -- if only it had a decent text editor.

But seriously: Emacs is a robust and extensible text-editing environment that has many, many additions designed into it, from compiling and debugging interfaces to e-mail, games, and Eliza. Especially for those who write or code (or both) for a living, it's easy to start up several Emacs sessions in the morning, start working, and never execute another application all day, thus the name of this tutorial: Living in Emacs.

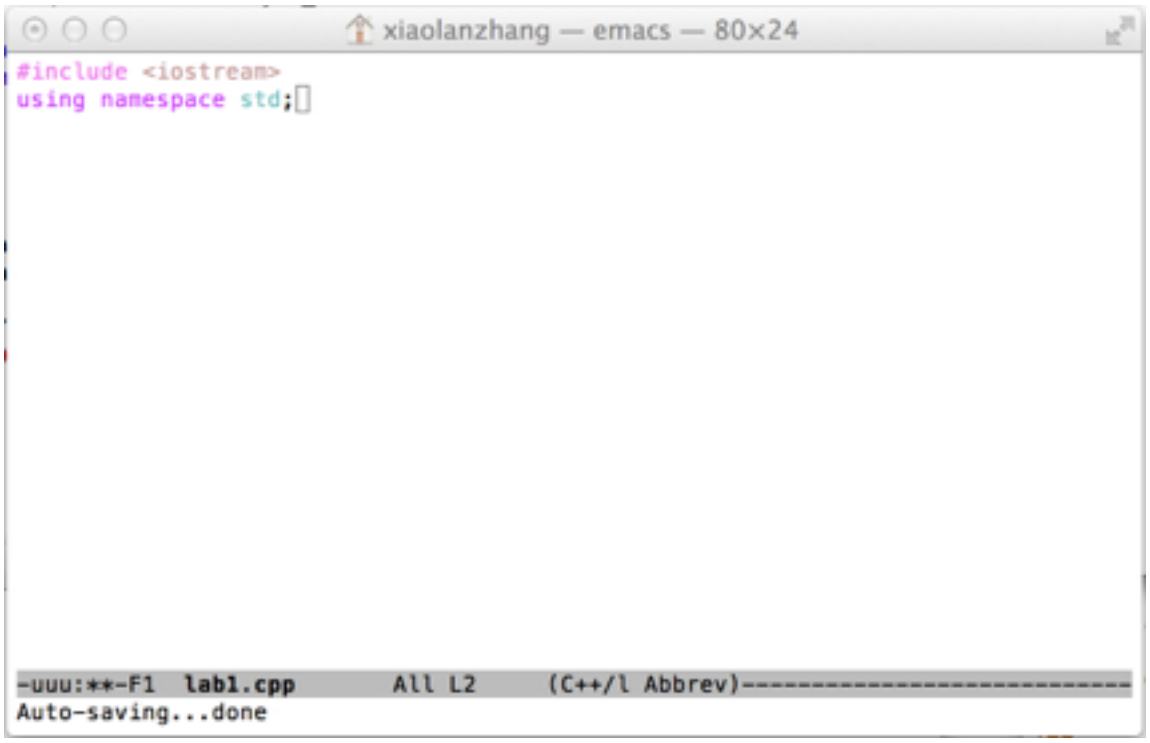
1. Starting out

In your terminal window, type the following command:

emacs lab1.cpp

There are three major sections to any Emacs or XEmacs screen: buffer(s), the status bar, and the mini-buffer at the bottom.

Figure 1. Emacs program layout



```
#include <iostream>
using namespace std;
```

```
-uuu:**-F1  lab1.cpp       All L2     (C++/l Abbrev)----------------------------
Auto-saving...done
```

The screenshot in Figure 1 is from the text-mode only version (as in a Mac OS terminal). In a X-enabled version of GNU Emacs, there will be GUI button menu on the top and the mouse-enabled scroll bar (in most cases).

The **main editing window** can be split into two or more windows, which can be views of the same buffer (file), or of different buffers. See Windows in Emacs for more details.

In the initial configuration, the editing window has a demarcation at the bottom by a **status bar** (also known as the **mode bar**). With multiple visible windows, each will have its own status bar. The status bar has indicators for whether the text in the buffer has changed, the file name associated with the name, the mode (shown as SGML in the screenshot), the current line number, and the position of the cursor as a percentage of the entire text. The mode indicates what type of text Emacs thinks it is working with and modifies the menus and functions accordingly.

The bottom line, which contains a [Autsaving…done] message in the screenshot, is called the **mini-buffer**. It's used to display partially-typed commands, the results of commands run, and occasionally to show minimal help.

2. Emacs keystroke conventions

Emacs documentation has a unique way of describing the keystrokes that are used to define actions. These are as follows:

C-<chr> == Control + character, pressed at the same time.
M-<chr> == Meta + character, pressed at the same time.

But what's Meta? Meta can be a dedicated key (sometimes so labeled), it might be the Alt key, or perhaps it doesn't even exist in the keymap that your system uses. That's okay; there is a fallback to Meta, which is to first press the Esc key and then the following character in turn (instead of together). This yields the same result as M-<chr>.

3. Commands and key-bindings

Emacs have many commands have names, like Buffer-menu-bury, backward-char, and forward-paragraph. And while they're logically arranged and named, there are over 1800 of them in my current installation, and that's one heck of a lot of typing.

That's why many of the commands are bound to key combinations, prefaced with the Control and Meta keys. To invoke a named command, start by typing M-x followed by the command name. To get a list of the key bindings, the long form command is M-x describe-bindings. Fortunately, there's a keybinding for that: C-h b

Type C-x o to swap to the listing window, C-s to do an incremental search, C-x o to switch back to your working window, and C-x 1 to close all windows except for the current buffer. Give that a try, and have a look at some of those commands -- there are about 600 or so that have key-bindings. Also, don't worry about the commands we used in this quick side trip, as we'll revisit all of them in turn later in the tutorial.

4. First instructions

Quitting: When I first started using Emacs, I found that I would get lost someplace
in the documentation, or in a welter of buffers that I was sure I hadn't opened myself, and so on. At that point, all I wanted to do was exit the system so that I could start over again and figure out where I went wrong.

Here's the sequence you type to exit Emacs: C-x C-c .

The keystroke convention that you saw above means to press Ctrl + x, followed by Ctrl + c . If you made any changes in any open files, then Emacs will prompt you, for example:
I'll reply y to any such prompts if I've made changes I care about, or press the ! to simply proceed with quitting, nothing saved.

To open an existing file after Emacs is started, type C-x C-f to find a file and load it into a buffer.

On the other hand, I most often want to save the work I've done and then continue typing. **So to save my work and continue**, the keystroke combination is C-x C-s .

5. Common text operations

Inserting text

Emacs is very easy in one important sense. No need to get into an insert mode or exit from any special command mode -- just type and you're inserting text. Let's repeat one thing here: save your work, early and often, with the **Save Buffer command, C-x C-s** .

Did you enjoy that? Now take a deep breath, and let's dive in to deleting text.

Basic deleting and undo

There are two different ways to delete text. First we'll address the first: Character deletion. Single characters are deleted in the manner to which you are likely already accustomed: by using the Delete key or the Backspace key.

Delete, at least, has an Emacs equivalent: C-d deletes the character under the cursor. To undo character deletion, use the C-x u command or the real shorthand, C-_. The latter is easier for multiple undos. Practice these operations just a bit now to start training your fingers in Emacs. Note: Some of the documentation I have read indicates that the Delete key should delete backwards (the backspace or ^H equivalent) and C-d takes the place of Delete. This depends on your operating setup and terminal configuration.

Deleted characters are only saved in a buffer for undo, and you can only reach those modifications by undoing all that's changed since the deletion. The more “advanced" form of deletion, for multi-character regions, is saved to a different structure as well, and we'll look at that next.

Emacs cut and paste

Here are the commands you need for deleting larger blocks (it's called "killing"):

Key-binding	Action (command)
M-d	kill-word
M-Delete	backward-kill-word
M-k	kill-sentence
C-x	Delete backward-kill-sentence
C-k	kill-line

C-k has a bit of a trick to it. Used once, it kills the text on the line but not the newline character. That takes a second C-k. There are also commands to kill paragraphs, kill-paragraph and backward-kill-paragraph, although key bindings don't exist for those.

So where does your deleted stuff go? Into the **kill ring**, of course. Multiple sequential deletes (for instance, repeating C-k several times) goes into the kill ring as a block, which is very handy.

Next we'll look at accessing that data.The kill ring is so called because it stores deleted text larger than a single character. Also, it can be accessed sequentially, from the latest back to the first item deleted during the editing session, and then it wraps back to the most recent again. Thus, it is a ring, topologically.

Type **C-y to yank the most recent block**. Repeating C-y merely yanks that block again.
To get at the older "killed" items, type C-y first, and you'll see the most recent block. Then, type M-y to step back through the kill ring. Each step replaces the prior yank. Give it a try now -- it's really quite handy.

The universal argument

The command universal-argument, with a key-binding of C-u, can be used as a prefix for a great number of other actions, including many of the delete commands I've shown you in the previous panels.

For example, typing C-u 6 C-k kills three lines. Yes, that's three lines, not six. Remember that with kill-line, the text on the line and the newline are done separately. Not hard to get your head around, once you've used it a few times.

Without a numerical argument, universal-argument defaults to a count of 4.

Basic operations in review

Here's a table of all the commands and their key-bindings discussed in this section. Give them a glance and make sure you know what they are. Practice with these briefly to gain more familiarity with the actions. First off, just type in the main window to insert text.

Key-binding	**Action (command)**
C-g (Esc Esc Esc)	keyboard-quit to get out of a command that's been started
Backspace	backward-delete-char
Delete (C-d)	delete-char
C-x u (C-_)	advertised-undo
M-d	kill-word
M-Delete	backward-kill-word
M-k	kill-sentence
C-x	Delete backward-kill-sentence
C-k	kill-line
C-y	yank is the paste equivalent
M-y	Traverse the kill ring, must follow C-y
C-u, C-u N	universal-argument, adds count prefix to commands

6. Cursor navigation in Emacs

Running Emacs in a GUI environment means you can use a mouse or directional keys like the Up and Down arrows and the Home and End keys to move the cursor around in a document. However, I'm going to review the native navigation scheme for Emacs, since this is the only method that's guaranteed to work, whether you're on a dial-up line from a terminal, accessing a machine via a console or SSH connection, or any of myriad other ways.

The native key navigation has the additional advantage of keeping your hands on the keyboard, where they belong, both for productivity and ergonomic reasons. I find that the context switch between keyboard and mouse costs me about 10% productivity when I'm using a tool in GUI mode.

Fire up Emacs as before (type emacs practice1.txt), and type a few lines into the initial window that you're presented with.

Little steps

Emacs occasionally uses character mnemonics to assist you, as your fingers learn the commands without conscious effort. Just remember Previous, Next, Forward, and Back. The first letter of each is your motion key.

C-f advances the cursor one character, while C-b moves it back one character. Note that this includes wrapping from line to line. C-n moves to the next line, while C-p moves the cursor up one line. Where possible, the vertical motion retains the column. However, if the next or previous line is shorter than the current cursor column, the cursor will automatically move to the end of the new line. Should you then continue onto a longer line, the cursor will return to the "original" column, in the new line.

Words, lines, and sentences

To move from word to word, Forward and Back still guide you, using the Meta key instead of the Control key. Note that words are defined as contiguous spans of letters and numbers. Punctuation counts as whitespace for word movement purposes. Try each of these commands several times as we go over them. M-f moves the cursor forward one word, while M-b moves back one word.

The mnemonic guidance starts to crumble a bit as we head into more line operations, where the 'a' and 'e' keys are beginning and end respectively. C-a takes you to the first column in the current line, and C-e takes you to the line's end.

At least we get to keep the same characters for stepping through sentences. Typing M-a takes us backward to the beginning of the current sentence (or the previous sentence if the cursor is at a sentence start to begin with). M-e moves forward in the same manner, relative to sentence ends.

Sentences are defined by punctuation and either a carriage return or two spaces. Depending on the text, the results might not always yield true sentence steps, but something closer to paragraphs.

Taking big steps

Moving one screen at a time is a handy operation, and here are the commands to accomplish that.

C-v scrolls the text forward one screen, and M-v backwards. Conveniently, there's a two-line overlap that makes it easier to retain your context. Additionally, typing C-l (that's a lowercase 'L') re-centers the window around the current cursor location.

Finally, to get to the beginning or end of the buffer, use the following keystrokes: M-< takes you up to the top, and M-> to the bottom. Those really are < and >, so you will need to use the shift key.

Cursor movement crib notes

Key-binding	**Action (command)**
C-f	**forward-char**
C-b	**backward-char**
C-n	**next-line**
C-p	**previous-line**
M-f	**forward-word**
M-b	**backward-word**
C-a	**beginning-of-line**
C-e	**end-of-line**
M-a	**backward-sentence**

M-e	**forward-sentence**
C-v	**scroll-up**
M-v	**scroll-down**
C-l	**re-center**

Practice these in the test document and keep using them. I found that I had to force myself not to use the cursor keys or the mouse for a while. By keeping my fingers on the home row and consciously using these commands, I was soon navigating through each file's buffer with ease.

7. Search and replace

Incremental searches

Incremental searches are one of my favorite features in Emacs. These start matching in the text immediately when you start typing. The advantage is that often you don't have to type the whole word before you've completed the search.

A standard forward incremental search is **initiated with the C-s command.** Searching backwards from the cursor position is accomplished with the C-r (isearch-backward) command. There are a variety of in-search commands available; you can get a complete description by typing:

```
C-h d isearch-forward
```

Highlights include incrementing through the matches by typing C-s for forward or C-r for backward steps. Also, press Enter or C-g to terminate the search when you've reached your goal.

Try using incremental search now. Position the cursor at the beginning of the practice document and type C-s. A prompt appears in the mini-buffer -- I-search:. Then, slowly type the letters of the word you're going to search for. As you add each letter, a highlighted area proceeds through your buffer, showing the first match for the part you've typed so far. In the screen fragment below, you can see the first match bounded in magenta, with the next potential match in light green.

Replacing text

There are two basic types of replace commands in Emacs. The first is an unconditional replace, based either on string or regular expression specification. There is no key-binding by default (I must therefore conclude that it's not regarded as significant), but it can be accessed by typing M-X replace-string (or M-x replace-regexp). This is followed by the target string/expression and the replacement string. Replacement is unconditional and forward from the cursor location only. The second command, query-replace, is bound to M-% (another shifted keystroke). After typing in the target and replacement strings at the prompts in the mini-buffer, each match in turn is highlighted, and you're prompted for the action to take. Pressing ? displays the complete list of possibilities here. The most common are 'y' to replace and continue, 'n' to skip and continue, 'q' to quit, and ! to replace all the remaining matches unconditionally.

Search and replace summary

Here's the table summarizing the fundamental search and replace capabilities of Emacs that we've covered. Remember that you can get detailed help for any command, with or without a key-binding, by typing C-h d command-name .

Key-binding	Action (command)
C-s	isearch-forward
C-r	isearch-backward
<find>	search-forward
Esc C-s	isearch-forward-regexp
Esc C-r	isearch-backward-regexp
n/a	replace-string
M-%	query-replace