The Woodnotes Guide to Emacs for Writers

Randall Wood (www.therandymon.com)

March 31, 2011

Contents

1 Introduction

1.1 License and Version History

 Please send comments, criticisms, and corrections to me at the email address found at my website. Enjoy this guide; I enjoyed creating it.

[1] http://creativecommons.org/licenses/by-nc-sa/2.5/

- 31 March 2011: Added Typing Special Characters, clarified keybindings, minor edits.
- 25 December 2010: Major rewrite and reorganization, much more concise. Also improved readability, updated information to reflect improvements made in emacs versions 22 and 23, added transposition and and case changing.
- 4 July 2007: Added info on changing colors and font display
- 6 October 2006: Added licensing information and this version history. Over 2000 people have downloaded this guide!
- 2 March 2006: second update incorporating comments
- 1 January 2005: Original version, courtesy of a cold spell in Washington DC

1.2 Introduction: Why a Text Editor instead of a Word Processor?

For many kinds of documents word processors are not the best tool for the job: long documents, documents like academic works that require careful organization, fiction and other long works of text all suffer at the hands of word processors. For that matter, only plain text can assure total compatibility between systems, as it is the lowest common denominator.

If you write fiction, poems, lengthy emails, or organized, long works, you may be well served by a text editor over a word processor. Word processors distract you with usually unnecessary options. Because your publisher is going to do the formatting, you will probably be requested to send in a document devoid of any mark up (bold, italics, etc.). And word processors can be distracting in their incessant highlighting of misspelled words that beg to be corrected before moving on – thus interrupting your train of thought. Word processors make you focus too much on the document as a document and distract you from what you should be doing, writing – being creative – producing text. Why worry about page breaks during the writing process? You should be dealing with that at the end, or not at all. In sum, text editors allow you to concentrate on writing, not formatting.[2]

Because text is so important to Unix, a tremendous number of powerful text editors are available to you, nearly always without charge. This document describes how to use one of them – emacs – to write. Its power in the hands of computer programmers is legendary. But it's a valuable ally for authors as well.

[2] If you truly need formatting for a long and complex document, you will probably be better off with a more powerful publishing tool than your typical word processor anyway.

2 Setting Up

2.1 Emacs on Linux/Unix, Mac OSX, and Windows

On Linux and Unix emacs is either already installed on your machine, or available via your software manager. It consists of several interlocking components though, and it's possible to install only the console version, missing the GUI (graphical) version. There's also xemacs, which for a time was far ahead of emacs in usability but has since fallen somewhat out of favor. Its menus make a lot more sense than emacs', in my opinion, but it's no longer very good looking, and the instructions here won't fully apply.

Mac OSX is Unix under the hood, so it runs emacs natively, and the console version, not the GUI version, is even provided free with new Macs. Open a new Terminal window (look in your applications folder) and enter the command `emacs` at the prompt, and emacs will start. There are other options as well. Enrico Franconi[3] has developed a fully carbonized version of emacs which provides the best of all possible unions between emacs and the Mac OSX operating system. The Mac menubar overhead takes the emacs menu, the program interacts perfectly with the Mac clipboard, and you can launch it directly from the OSX Dock. Moreover, the Apple Command key ("flower") serves as "Alt" (shown as `M-` in this document), which is far more convenient than reaching for the Escape key as you have to do otherwise.

There's also Aquamacs, a version of emacs that tries to reach a compromise with Apple's desktop, using Macintosh native file dialogs and more. It's comfortable software in a lot of ways, and the first application I reach for on Mac OSX when I want emacs.

For Windows, There is a native version of emacs available from GNU's Savannah server.[4]

2.2 X or Console?

This is a question for Unix/Linux users, and you probably already know the answer because it's a matter of personal preference, but it's worth looking into. Emacs was a console application for years, but developed an optional graphical interface (GUI) because users demanded one, and now you can use whichever you like best, or both.

If the graphical emacs has been installed, emacs will run with clickable menus and lots of mouse support; in the console version you will access the menus through keystrokes, not by clicking, but all the same functionality will be there. If you are working in a virtual terminal and want to specify the emacs you launch to be the console version, launch it with `emacs -nw`. This is actually my favorite way, as I can run emacs in a transparent terminal with a cool background, and I don't want or like the mouse support much anyway.

[3] `www.inf.unibz.it/~franconi/mac-emacs`
[4] `http://ftp.gnu.org/gnu/emacs/windows/`

3 The Basics

3.1 Some Vocabulary

Emacsen[5] have a steeper learning curve than your average software because they don't use the vocabulary words you'd expect them to and therefore it takes a longer time to find what you're looking for in the manual or learn the options you'd like to understand. You may very well know what you want but not know how to find it. A simple vocabulary lesson will set you a long way forward in your effort to learn to use emacs.

Frames are what any other program would call "windows." An emacs frame can be simply another view of the same document or show different documents.

Buffers: When you load a document into memory, your work resides in a memory buffer, while the file remains unchanged on the disk. Then when you save your work, the buffer is written to that file.

Window: You can divide your screen into sections called "windows," each of which can view a different buffer or different parts of the same buffer.

Filling is what other programs call "word wrap," sort of. To get a paragraph of text to "wrap" you must essentially insert a return character at the end of every line at a certain position, say every 80th character or so to have paragraphs formatted 80 characters wide.

Kill means to remove text. Everyone else calls it "cut."

Yank means to insert previously removed text, i.e "paste."

Copy to Kill Ring is the equivalent of "copying" text in other applications.

Figure 1 shows a summary of emacs lingo and their equivalents for other software packages.

Other Software	Emacs
Window	Frame
File	Buffer
Tab/Pane	Window
Paste	Yank
Formatting/Justification	Filling
Cut/Paste	Kill/Yank

Figure 1: Emacs Vocabulary and meaning

First of all, a word about notation: emacs commands all start with the control key or the alt key. The command Control-X is shown as follows: `C-x` and the command alt-X

[5]Emacs and its variants (see, you're using new vocabulary already!)

is shown as follows: `M-x` (the M stands for "meta" and goes back to the days before the alt-key). Some commands involve several steps, like the following, which sets the margin to 20 characters: `C-u 20 C-x f`. So hit control-u, type the number 20, then hit control-x, release, and strike the f key. The status bar at the bottom of the screen shows your progress. If you screw up half way, hit `control-g` to cancel the command (you can't edit the command half way, you have to start over). Now that you understand the notation, you'll understand `C-x C-c`, which means "exit emacs."

You'll feel more comfortable exploring emacs once you know how to undo mistakes. The command is `C-x u` and emacs remembers a long history of your previous commands so it can undo a lot of mistakes. Figure 2 summarizes these basic commands.

Command	Key Sequence
Cancel	C-g
Quit emacs	C-x C-c
Undo	C-x u

Figure 2: The Basics

3.2 Emacs Commands

The "mac" in "emacs" came from the word "macros." Every single command available to you, and commands you write yourself (see Section 8.1: Macros below) is a function written in a programming language called emacs-lisp. They all have names like canonically-space-region or indent-region or ispell-buffer. Many of those commands are associated with keyboard shortcuts like `C-n` (to move the cursor down one line) but they also have a long name as well (in this case `C-n` is the command next-line). Not all of the commands available to you have a keyboard shortcut; those that don't are accessed by hitting `M-x` and their long name. For example, type `M-x ispell-buffer` to begin spell checking the entire document. Once you hit `M-x` you can type just `isp` and hit tab, and emacs will try to complete the command with the options available to it. In this case it will get as far as ispell- because there are several commands whose names start with that sequence of letters. Continue by typing `bu` and hitting tab. Emacs will now complete the command: `ispell-buffer`. Hit return and the spell checking will begin.

In the rest of this document, remember that commands that don't have a shortcut like `C-t` (transpose characters) can be accessed by typing M-x plus the long name of the command, so `M-x transpose-characters` and hitting return.

You'll notice as you get more familiar with emacs that commands come in basically three flavors: those that begin with C-x, those that begin with C-c, and those that begin with M-x. C-x is reserved for the most common commands and particularly those that involve reading in and saving buffers, so `C-x C-f` to "find" a file, `C-x C-c` to quit, and so on. Commands that are less frequently used get relegated to `C-c`, which you'll notice is a little more of a stretch for your finger to reach. `M-x`, as explained above, is used to access commands by their long names. While there are a couple of other key combinations, like `C-h` for commands related to the help system, these are few in number.

3.3 Files (Opening, Saving, Printing, etc.)

Remember, a file is what's stored on disk. Once you read it into emacs to begin editing, we refer to it as a buffer, because it's been stored in memory. In general, opening, saving, and printing files is straightforward. A couple of notes: To "open" a file and to create a new file are the same thing: emacs will try to find the file you request and if it doesn't exist it will simply create a new, empty file for you. When you "save as" by typing `C-x C-w`, emacs will save the file under a new name and then continue to edit the newly-renamed file. This is what most programs do, but it's worth mentioning because there are some programs that simply save your current work to disk under a new name but continue to edit the original file (called "save a copy as").

Inserting a file at the present cursor position is extremely useful. I use it to send templated emails. I start off with a personalized introduction, then insert a file which contains one of several templates, edit as necessary, and send. You'll find other uses for this feature as you go.

The basic commands are shown in Figure 3.

Keystroke	Command/Function
C-x C-f	Open ("Find") a file
C-x C-s	Save
C-x s	Save some or all files to disk
C-x C-w	Save as
C-x i	Insert another file into current buffer
C-x C-v	Replace this buffer with another file

Figure 3: File Commands

Finally, there are several ways to print, but one of the nicest is `M-x ps-print-buffer-with-faces`, which will print your text buffer to the printer with fonts, syntax highlighting, and all. It generates very attractive print-outs of your work.

3.4 Navigating

Once you learn the keyboard shortcuts, you'll find they are far quicker than reaching for the mouse. Press `C-f` to move forward and `C-b` to move backward; `C-n` to move down to the next line and `C-p` to move up to the previous.

You can repeat any number of times by prefixing an argument with the command Control-u as follows. Let's say you want to move forward 8 characters. Enter `control-u 8 control-f` all in a row. We'll see the control-u command later in this document for other commands that require a number, like setting margins to a certain number or characters wide and so on. Emacs has commands to move the cursor by other units as well: by a word, to the beginning or end of the current line, by a sentence, and by a paragraph, as shown in Figure 4:

Entity to Move Over	Backward	Forward
Character	C-b	C-f
Word	M-b	M-f
Line	C-p	C-n
Sentence	M-a	M-e
Paragraph	M-{	M-}
Page	C-x	C-x]
Beginning/End of Line	C-a	C-e
Beginning/End of Buffer	M-<	M->

Figure 4: Cursor Movement

3.5 Scrolling

The two most important commands are `C-v` to scroll down one screenful and `M-v` to scroll up one screenful. The cursor moves with you. And no matter where your cursor is in the document you can scroll that point up to the center of the screen by with `C-l`. Together they are an easy way to navigate quickly up and down through the document. But there's a quicker way still to get where you're going: incremental searching. We'll look at that trick in section 3.9.

Scroll Direction	Backward	Forward
Previous/Next Screen	M-v	C-v
Scroll Left/Right	C-x <	C-x >
Scroll Current Line to Center of Screen	C-u C-l or C-l	
Scroll Other Window	M-C v	

Figure 5: Scrolling

3.6 Bookmarks

As you write, you may find it convenient to place a bookmark at certain points in your text so you can conveniently return at some future point. Emacs allows you to set, remove, and name bookmarks.

If you're using the graphical (GUI) version of emacs, the bookmarks functions are available to you under the Edit–>Bookmarks menu. Otherwise, remember the following commands (the lisp-function is available to you by hitting `M-x` and the name of the function, i.e `M-x bookmark-jump`).

3.7 Selecting Text ("Regions")

Most anything you do, including cutting and pasting (see section 3.8 below) involves selecting or highlighting an area of text. To do so, you position the cursor somewhere

Action	Keystroke	Lisp Function
Set Bookmark	C-x r m	(bookmark-set)
Jump to Bookmark		(bookmark-jump)
Delete Bookmark		(bookmark-delete)

Figure 6: Bookmarks

and set a mark, then move to somewhere else and define everything between the mark and your current position as the region. Once you've selected the region you can go on to cut it, format it, etc.

So, put the cursor somewhere and press `C-space`. The status bar at the bottom of the screen should indicate "mark set." Now using the scrolling and cursor movement commands described in section 3.4 to get to where you want. Everything between your current position and the mark should be highlighted and is now considered the region. The next command will affect the entire region. Two quick shortcuts: `M-<` and `M->` will select from the cursor point to the beginning/end of the buffer, respectively.

If for some reason, the region does not get highlighted as you select it, it means transient-mark-mode has been toggled off. Toggle it back on by entering `M-x transient-mark-mode`. Apparently, not everyone likes to see the highlighting, though I certainly do.

3.8 Cutting and Pasting (Killing and Yanking)

This is the most obvious example of how emacs doesn't follow naming conventions used by most other software. Killing and Yanking, once you get used to the new vocabulary, does what you'd expect it to, but emacs provides some other tricks that are useful to writers. First of all, emacs remembers more than one thing cut ("killed") and keeps them in a list called the kill ring. You can later paste ("yank") not just the most recent thing killed but previous things as well. Simply hit C-w to "kill" something (i.e. "cut" it). To "yank" it, hit C-y (i.e. "paste"). To copy something to the kill ring (i.e. "copy"), use the Alt key instead of the Control key, that is M-w. The basics are shown in Figures 7 and 8.

Entity to Kill	Backward	Forward
Character	DEL	C-d
Word	M-DEL	M-d
To end of line	C-k	
Sentence	C-x Del	M-k
Entire line	M-x kill-entire-line	

Figure 7: Killing (cutting)

Let's look for a moment at how the kill ring works. Imagine an immense list of everything you've killed during this session that you can cycle through. When you "yank" some text, the most recent item is what you get. But if you immediately hit M-y, that text is replaced with the previous item. Hit M-y again to replace that with the item previous

to that, and so on until you get what you want. It's called a kill ring because you cycle through all the items, eventually returning to the most recent item killed.

Keystroke	Command/Function
C-w	Kill ("cut")
M-w	Copy to Kill ring ("copy")
C-y	Yank ("paste")

Figure 8: Killing and Yanking

Lastly, you can also kill things selectively – a word here, a sentence there – and accumulate them as you go, kind of like a selective harvest of your text. You may not use these tricks frequently, but they're highly convenient when you need them:

Command	Purpose
append-to-buffer	append region to particular buffer
prepend-to-buffer	add the region to the beginning of a particular buffer
copy-to-buffer	replace specified buffer with contents of region
append-to-file	append region to contents of a specified file

Figure 9: Accumulating Text in Buffers

3.9 Searching and Replacing

Searching for text is important all throughout the processing of writing, but it's also a useful way to navigate a document too, if you can think of words specific to certain areas of your text that will allow you to pinpoint it. Emacs provides a very powerful way to search your document, and several additional search and replace mechanisms of use to writers.

The one you'll use most frequently is called incremental searching. When you press `C-s` emacs will prompt you for what you want to search for. It will then search as you type from the cursor position to the end of the document. But as you continue typing it will add those letters to the search. An example is worth a thousand words. Let's say you're searching for the word "iconoclastic" in your document. Hit `C-s` and start typing "i-c-o-." As you type "i" emacs will highlight the next word that starts with "i," but as you type "ico" it will highlight the first word that starts with "ico," building as you type until finally you've typed in the whole word and emacs is highlighting the next occurrence of the word "iconoclastic." If that's the position in the buffer you want to skip to, press `enter` at this point, or `C-g` to remain where you were when you started the search. `C-r` performs an incremental search backwards from the cursor.

Type `M-%` to begin the query replace process. This is the equivalent of search and replace in other software. Emacs will ask you at each word if you'd like to replace that occurrence of the word or no, and you can answer yes, no, yes from now on, cancel, and so on.

One limitation of regular searches in documents that have been formatted (filled) is that if the two words are separated by a newline, the incremental search function won't find

them. For these cases, use the word search command: `M-s w`. It will find a phrase regardless of punctuation or spaces between the words.

Command/Function	Keystroke
Incremental search forward	C-s
Incremental search backward	C-r
Query replace	M-%
Word search forward	M-s w

Figure 10: Searching and Replacing

4 Foreign Languages and Foreign Characters

4.1 Occasional Diacriticals

It's not immediately obvious how to insert characters from foreign alphabets and symbols into your text, but emacs is more linguistically-aware than a lot of software, so it should come as no surprise that it's possible.[6] In fact, you can use emacs to compose texts in Asian alphabets (e.g. Chinese, Korean), right-to-left alphabets (e.g. Arabic, Hebrew), and insert any character in the unicode definition.

If you are mostly writing English, with the occasional odd foreign character, it's easiest to use C-x 8. For example C-x 8 Y produces the Yen symbol (¥), C-x 8 L produces the pound symbol (£), and C-x 8 o produces the degree symbol (°). There are about two dozen characters easily available through this method. Type `C-x 8 C-h` for a list of them (or type `C-h b` for a list of all keybindings).

Keystroke	Produces	Meaning	Keystroke	Produces
C-x 8 Y	¥	Yen	C-x 8 ´ e	é
C-x 8 L	£	Pounds	C-x 8 ` o	ò
C-x 8 o	°	Degrees	C-x 8 ^ o	ô
C-x 8 R	®	Registered	C-x 8 ~ n	ñ
C-x 8 C	©	Copyright	C-x 8 , c	ç
C-x 8 c	¢	Cents	C-x 8 / o	ø
C-x 8 S	§	Section	C-x 8 "u	ü
C-x 8 P	¶	Paragraph	C-x 8 "s	ß
C-x 8 1 2	$\frac{1}{2}$	One Half	C-x 8 /e	æ

Figure 11: Some Commonly Used Special Characters

[6] Many thanks to Xah Lee (`http://xahlee.org/emacs/`) for excellent base material for this section.

4.2 Writing in a Foreign Alphabet

If however, you are going to write a text in Spanish, French, Turkish, or some alphabet that makes regular use of diacritical marks, you are better off changing the input method. The easiest way to do this is either `C-x RETURN C-\` or `M-x set-input-method`. Hit tab to see the full gamut of input methods available to you, from Georgian to Tibetan to Dvorak and beyond. In these cases, emacs will assume you're using a keyboard where certain keys correspond to certain particular characters in that alphabet.[7] Assuming you just want to type some French or Turkish, something like latin1-postfix or latin1-prefix should do the trick. In both cases, you create a character like à by using the ` and `a` keys on your keyboard: for latin-1-prefix you type the symbol before the letter and in latin-1-postfix you type the symbol after the letter. This is equivalent to the "dead keys" keyboard layouts provided by the keyboard settings in most Linux desktops (KDE, Gnome, XFCE).

If you are going to switch between two input methods (i.e. keyboard layouts), use `C-x RETURN C-\` the first time to make the first switch. Subsequently, toggle back and forth between the two layouts by using `C-\`.

4.3 Inserting Special Characters

Emacs uses unicode internally, and can produce any character in the unicode definition. You may, however, not be able to see that character on your screen because your Linux distribution doesn't have the proper font to show it. If you know the character by its unicode name, you can insert it that way. Type `C-x 8 RETURN` and then begin typing its name. Emacs will load the list of known unicode characters and try to guess. For example, `C-x 8 RETURN BLACK CLUB SUIT` will produce ♣.

If you know the character's code, you can enter it the same way. The double dagger (‡) is unicode U+2021, so `C-x RETURN 2021 RETURN` will produce it. Wikipedia has a decent but incomplete list of unicode characters by number[8] but the source[9] is more complete, and affords you a sense of the awesome breadth of the unicode standard.

A much more cumbersome method is to change to the 'ucs' input-method before inputting a string of characters by their unicode number. Type `C-x RET C-\` and when prompted, choose the 'ucs' (direct unicode) method. Now type `u` plus the number of the character (`u2021` in this example of the double dagger); repeat for the next characters.

If you make use of some unusual symbols fairly regularly, it is worthwhile to bind those characters to a keystroke. See section 8.2 for details on how to do so.

---

[7] I haven't got a Georgian or Tibetan keyboard to test this on, sadly, so can't confirm this hypothesis.

[8] `en.wikipedia.org/wiki/List_of_Unicode_characters`

[9] `www.unicode.org/charts/`

5 Formatting Your Text

5.1 Word wrap

Before version 22 word wrap was a bit of a hassle, but things are much better now, so upgrade if you can.[10] Word wrap comes in two forms: soft wrap and hard wrap. Hard wrap means that at the end of every line a "newline" character is inserted. Most plain text email is sent this way. If you have 80 character wide paragraphs and want them to be 120 characters wide, you have to reformat. Soft wrap means the program recognizes the width of the window on your screen and reformats the words to fit the window, without inserting any newline characters. If you resize the window, the words adjust automatically.

Unless you tell it do otherwise, emacs will let lines wrap around the screen, but make no effort to break the line between words; you'll see a little arrow right in the middle of a word. This is annoying. Better is to enable "long lines" mode (`M-x long-lines-mode`, which permits emacs to leave words intact. The lines will wrap at the margin set with the `C-x f` command. `C-u 80 C-x f` sets the width to 80 characters, for example of your paragraph but does not reformat the paragraph. M-q reformats the paragraph.

For hard-wrapped lines (useful, for example, if you're writing a LaTeX document), hit `C-u 120 C-x f` to set the margin, and type `M-x auto-fill-mode` to toggle auto-fill mode on (check the status bar at the bottom of the screen to see if it's on: look for the word "fill" in the mode line). Now start typing. Your paragraphs will be hard wrapped at 120 characters, the width of your screen. Now if you go back to edit your work, the paragraph will be out of whack. Hit `M-q` to reformat the paragraph. If you later decide you want the paragraph to be 72 characters wide again, you can hit `C-u 72 C-x f` to set the new margin and `M-q` to reformat it.

There are two other useful commands available to you if you've selected a region you'd like to format. The command `fill-individual paragraphs` reformats each paragraph in the region. This is probably what you want if you want to globally change all the paragraphs in your document from 72 to 85 characters wide, for example. The command `fill-region-as-paragraph` will take all the fragments of text in your region and make them into a single paragraph, removing extraneous blank lines and double spaces, etc. – a very handy way to reformat hacked-up text.

5.2 Reformatting Hard Wrapped Documents

If you have a document that has already been hard-wrapped, getting rid of all those newline characters and going back to soft wrapping is not quite intuitive either. One place

[10] For versions prior to 22 you'll have to find the package on the Internet and install it with the rest of the lisp code on your system. Where you install it might not be obvious. On my Linux system it was a matter of copying the file to /usr/share/emacs/21.2/site-lisp/ and setting the permissions to -rw-r–r–. I added the following line to my .emacs file: `(autoload 'longlines-mode "longlines.el" "Minor mode for editing long lines." t)` Then when you need it, simply issue the command `M-x longlines-mode`

where you'll run into this is reformatting plain text email for use in another program. Before you can do much with the text you need to get rid of the carriage return at the end of each line. There are two easy ways to do this.

The first way is the most simple: First, set your fill-column variable to some huge number greater than the probable maximum number of characters in a single paragraph, like 10,000. Then select the whole document and invoke "fill-individual-paragraphs." Remember, to set your fill-paragraph variable the key binding is `C-x f`. So to set it to 10,000 you'd hit `C-u 10000 C-x f`.

The second way is easier, but may only work if you're working on Linux or Unix, which has the following tool available. Simply highlight the area or the whole document, and type `M-1 M-| fmt -w2000` This pipes the text to the GNU fmt ("format") command with a paragraph width of 2000 characters. If you can remember the keystroke, this is the most elegant way to do it.[11]

If you're going to use this command frequently it may make sense to define it as a macro and bind a keystroke to it so you can evoke this function with a single keystroke (see section 8.1 for more info on macros and section 8.2 for more info on binding keys). This is the code you would add to your .emacs file:

```
(defun fix-screwed-up-paragraphs(beg end)
   (interactive "r")
   (shell-command-on-region beg end "fmt -w2000" nil t))
```

5.3 Transposing Letters/Words/Lines

`C-t` will transpose your current character with the character previous and `M-t` will do the same with words. `C-x C-t` will do the same with lines (remember a line is not the same as a sentence). `M-T` will transpose your current line with the line above it. Emacs offers other transpose commands but I rarely use any of them; find them by typing M-x apropos, hitting return, and then typing "transpose."

Keystroke	Command/Function
C-t	Transpose two characters
M-t	Transpose two words
C-x C-t	Transpose two lines

Figure 12: Transposing

5.4 Cleaning Up Spacing

The commands presented here complement the filling techniques described above and provide some additional functionality as well that's useful for writers. `C-o` is one of the

[11] Thanks to Jerry Sievers for the first technique, and Rod (author of *Linux for Non-Geeks – Clear-eyed Answers for Practical Consumers*) for the second technique. Thanks to Marc Girod for the lisp function

commands I'm most grateful for. It inserts a blank line beneath the cursor and forces all the rest of the text down from the position of the cursor. In other programs you have to hit return and then arrow your way back up to do this. `C-M-o` does the same, above the cursor.

Just as useful is the opposite: say you've got several blank lines between two paragraphs and you want to clean it up. Rather than manually deleting each line, just hit `C-x C-o` to remove all blank lines except one.

Two additional commands, `M-backslash` and `M-space` clean up space between words, the former removing all spaces and tabs thereby juxtaposing the two words, and the latter removing all spaces but one. Finally, invoke `M-x canonically-space-region` after selecting some text. It will remove all extraneous spaces so that there's one space between words and two after a period. If you've seriously mashed up your text, this is a quick way to put it back together. Shortcuts like these are the ones that give you the advantage over word processor-users. `M-^` will join your current line to the previous.[12]

Keystroke	Command/Function
C-o	Insert blank line
C-x C-o	Remove all blank lines above and below but one
M-\	Concatenate text to left and right of point
M-space	Remove all spaces except one
M-^	Join this line with the previous

Figure 13: Transposing, Joining, and Formatting

5.5 Changing Case

It's easy to select text and make it upper case, lower case, or title case. `M-l` converts to lower case the rest of the word starting at the cursor; `M-u` converts the rest of the word to upper case. `C-x C-l` and `C-x C-u` do the same thing, but for a region. The `upcase-initials-region` command is what word processors call title case, in which the first letter in each word is capitalized. Note that because the `M-a` keystroke navigates to the first word in a sentence, you can use it effectively in a macro that goes backward through a text, capitalizing the first word of each sentence.

Keystroke	Command/Function
M-l	downcase-word
M-u	upcase-word
C-x C-l	downcase region
C-x C-u	upcase region
(not bound)	upcase-initials-region

Figure 14: Changing Case

[12] I find this key combination cumbersome and use it a lot, so I like to bind it to another key, like `M-j`

5.6 End of Line Characters

If you frequently deal with text files created by Windows users, you will no doubt encounter the frustrating \M character littered throughout the text. Remember that Unix, Windows, and Macs prior to OS X all deal with the end of lines differently. Windows marks the end of a line with two characters – an end of line (\ n) and a carriage return (\ r). Unix just uses the end of line (\ n), and Macintosh just uses the carriage return (\ r). When you open a text file originally created in Windows, the \M characters represent left-over carriage returns emacs didn't know what to do with.

There is an easy way to get rid of them by just searching and replacing. Navigate to one of them, select it the way you would any other character or expression, and copy it using `M-w`. Then Hit `M-%` to begin a search and replace session. When emacs asks what to replace, hit `C-y` (yank). When emacs asks with what to replace the character, just hit `return`. Emacs will then remove all those \M characters. You can also type `C-q` ("enter a literal") followed by `C-m` to enter the end-of-line character directly, when asked.

6 Multiple Windows, Buffers, and Frames

Remember that what you would call "files" in other programs are "buffers" to emacs. That distinction becomes important when we start dealing with multiple buffers and introduce the concept of windows and frames (for a quick review look at figure 1).

First of all, let's say you begin working on a file called one.txt. But you want to work on two.txt as well, maybe because you're going to cut and paste text from one file to the other. So once you've got one.txt on your screen, type `C-x C-f two.txt`. The second file should be loaded into a buffer and that buffer should be the active one. What happened to one.txt? It's still in memory, but your window/frame is only presenting you one buffer, and it's two.txt. You can switch between the two buffers by hitting `C-x b`. Emacs will ask you "Switch to buffer (default one.txt). By simply pressing enter emacs will switch to one.txt. Alternatively, you can hit the tab key, and emacs will divide into an upper and lower section, and one section will show a list of all possible buffers. Some are special emacs buffers like logs. *Messages* is an example of one. You can type the first few letters of the buffer you want to switch to and emacs will auto-complete for you. When you've chosen the buffer you want to work on, press `enter`. By default emacs will assume you want to work on the buffer you were dealing with last, which means it's easy to switch between two buffers simply by hitting `C-x b` and `return`, accepting the default.

If you want to see a list of all buffers currently in use, type `C-x C-b`. You'll see a window listing all current buffers in use.

Once you've got two windows open though, how do you get rid of the second one? There are several ways.

- Make your current window the only one by hitting `C-x 1` (mnemonic: "one window"). The other window will disappear, though the buffer will still be open, just not displayed.

- Switch to the other window and kill the window. Switch by hitting `C-x o` (mnemonic: "other window"). The cursor will now be in the other buffer. Now kill it with `C-x 0`. The current window will disappear and the other one will occupy the full screen.

Let's look at some other ways you can deal with multiple buffers and windows. Hit `C-x 2` to divide the current window in two pieces, one upper and one lower (two vertical windows). Because emacs doesn't know what else to do, both windows will at first show the same buffer, so if you're currently working on one.txt and split the window, you'll have one.txt in both the top and the bottom windows, and both will reflect changes made in the other (think of them as viewports looking onto the same area of memory). This can be very useful when you want to have on the screen at the same time two different areas of your text.

Hit `C-x 3` to divide a window into two side-by-side (horizontal) windows. This can be handy to view side by side two buffers, for example to compare them or look for changes in one. But if you enable `follow-mode` the two side-by-side buffers will show consecutive sections of the same buffer, and scroll together as you move up and down. On modern, wide screen monitors, this is a great way to maximize how much of your text you can see at once.

Needless to say, you can have different buffers in each window. Start with one window and then using `C-x 2` divide it into two windows. Now switch to the other window with `C-x o` and then open up a new buffer with `C-x f` (to deal with a new file) or `C-x b` (to deal with an already opened file). Use `C-x k` to "kill" a buffer (close it permanently; emacs will ask if you want to save changes first) or C-x 0 to close a window without closing the buffer. Now is it clear why it's important to understand the difference between files, buffers, and windows?

A frame is what other programs call a window so be careful to distinguish with your vocabulary. An emacs frame is like a detached window, and like windows, just provide a viewport to a buffer, so you can add and delete additional frames without affecting buffers at all. Needless to say, the concept of a frame only works if you're using emacs in a graphical environment. If you're working at a virtual console, frames are not available to you.

Type `M-x find-file-other-frame`. Emacs will ask you for the name of the file and then open it up in a new frame. Or type `M-x new-frame` to display the current buffer in a new frame.

Keystroke	Command/Function
C-x b	Switch to other buffer
C-x C-b	Show the list of active buffers
C-x 2	Split the window vertically
C-x 3	Split the window horizontally
C-x k	Kill buffer (close the file)

Figure 15: Windows/Buffers/Frames

7 Spell Checking

Spell checking is one of several ways emacs interfaces well with other software to expand the tools available to you. Emacs spell checks via the ispell program, available on all Unix/Linux systems and on both Windows and Mac OSX as well (though note Mac OSX uses its own spell checking mechanism, and your ispell dictionary is separate from the others).

To spell check a word, simply hit `M-$` while the cursor is somewhere in or at the end of the word. Ispell will check the word and allow you to correct it, if necessary. To spell check the entire document, enter `M-x ispell-buffer`. You can add words to your dictionary as necessary as you go.

Flyspell mode is the equivalent of that check-as-you-go spell checking that some word processors use, and is one of the features that proves you can have in a text editor the same features you have in expensive, proprietary word processors. Enter `M-x flyspell-mode` to toggle the mode on or off. Flyspell mode uses the ispell program to spellcheck your document as you type and changes to a different color all the words that appear to be misspelled. It only checks what you type from the moment you toggle the mode on, however. If you've already typed quite a bit and would like to flyspell all the existing text, once you've toggled on flyspell-mode, enter `M-x flyspell-buffer` to have ispell look over your entire buffer for spelling errors. If you get tired of the distraction of misspelled words (it's often better to just do all your spell checking at the end of the day instead of as you go, so you don't get distracted) you can toggle flyspell mode back off again with the same command.

Lots of good dictionaries have been developed for ispell, all of which you can download for free via your Linux system's package manager. Enter `M-x ispell-change-dictionary` and change the dictionary ispell will use for the next command. This is very handy if you use emacs for email and find yourself communicating in more than one language.

8 Customizing your Environment

8.1 Macros

Macros are a great way to take the burden out of repetitive, annoying tasks. As mentioned above, the "mac" in emacs stands for "macros" and it's something emacs does very well. Simply hit `C-x (` to begin recording a macro, and `C-x )` when you're done. The status bar at the bottom of the screen will read "defining keyboard macro" as you type. Once the keyboard macro has been defined, it's an easy matter to invoke that macro throughout your typing session by typing `C-x e` and to re-invoke it by just striking e again as necessary. Or you can select a region and use `M-x apply-macro-to-region-lines` to edit only a certain portion of your text with the macro (`C-x C-k r`).

If you want to keep the macro available to you, save it and give it a name: Type `M-x name-last-kbd-macro` or `C-x C-k n`. You can edit that same macro by typing `M-x`

`edit-kbd-macro`. A new buffer will open with the commands that make up your macro for editing.

8.2 Keyboard Shortcuts

Emacsen are most frequently derided for difficult key combinations using one or more of shift/alt/control etc. You can create easier keystroke combinations to suit your needs by editing the .emacs file in your home directory. Add lines like the following:

```
(global-set-key [f1] 'goto-line)
(global-set-key[?\M-\l] 'next-line)
(global-set-key [?\C-x ?\w] 'beginning-of-buffer)
(add-hook 'text-mode-hook 'longlines-mode)
(global-unset-key (kbd "M-j")  )
(global-set-key  (kbd "M-j")  'join-line)
(global-set-key (kbd "M-*") "dookie")
```

The first three lines above set the F1 key to the command "goto-line," Alt-l to "next line" and C-x w to "beginning of buffer." Type C-h b to get a listing of all current key bindings. The fourth line instructs emacs to load the longlines.el package whenever it edits in text mode. The fifth and sixth lines remove the previous keybinding for Alt-J and reassign it to the command `join-line`. The seventh is an example of how you can bind a key stroke to a character you wouldn't normally find on your keyboard or a whole word you don't want to have to retype each time.

Note the many different styles for representing keystrokes. I find that in earlier versions of emacs the ?\M notation was necessary, while in versions 22 and up the (kbd) notation works better.

8.3 Fonts and Colors

It should come as no surprise that you can change many aspects of the emacs user interface. Here are some commands that help you customize your work environment:

The default emacs frame or window might be good enough for some, but you may get tired of black on white or simply prefer another layout like yellow on blue (like the good old Wordperfect for DOS days), or white on grey (for writing at night in low light conditions) to give your eyes a rest.

Set the foreground color and background colors with `M-x set-background-color` and `M-x set-foreground-color` respectively. As you press return, a buffer window will open up requesting you input the color you like. If you're working in X (as opposed to at a console) you have dozens of colors at your disposal, while on a console you have just eight. Start typing 'bl' for blue and then press tab and note how many blues are available. Same goes for most colors. For that matter, you can choose the cursor color and mouse color in the same way, namely `M-x set-cursor-color` and `M-x set-mouse-color`.

If you decide to make these your defaults, it's a matter of adding something like the following to your .emacs preferences file:

```
(set-foreground-color "white")
(set-cursor-color "red")
(set-mouse-color "goldenrod")
(set-background-color "black")
```

I have never had much of a problem with emacs' default font, and since writers of long works don't have much need for font changing, once you've chosen a good font you can stick with it. Remember, since this is a text file the font is only used for display on the screen anyway, not printing. As of emacs 22 you can select your font using the cool GTK+ font chooser dialog. But if you want to do it the old/difficult way, type `M-x set-default-font`, and when emacs prompts you for a font, hit tab to see what's available. Note which one works the best for you, and then add it to your .emacs file with something like the following:

```
(set-default-font
"-Misc-Fixed-Medium-R-Normal--15-140-75-75-C-90-ISO8859-1")
```

8.4 Default Window Parameters

Add the following to your .emacs file to set a default new window size if you constntly find yourself resizing them as they appear.

```
(setq default-frame-alist
 '(
; frame width and height
  (width    . 80)
  (height   . 40)
  )
)
```

8.5 Menus and Toolbars

You might be interested in the menubar and icon toolbar at the top of an emacs window, but personally I find the emacs menus unintuitive and the icon toolbar kind of superfluous, and use neither. Fortunately you can turn both off if you like using `M-x menu-bar-mode` (it's a toggle) and `M-x tool-bar-mode`.

To make these preferences permanent, add `(tool-bar-mode -1)` and `(menu-bar-mode -1)` to your emacs file, respectively.

8.6 Other Environment Settings

You'll be amazed by just how many aspects of emacs' behavior are customizable. But you just want to write, not spend all day customizing. In a nutshell, here are some aspects you might want to ensure are set to your liking and then never touch again. Add these to your .emacs file to set them and forget about them.

Put this in your .emacs	To achieve this effect
(setq inhibit-startup-message t)	Disables the splash screen
(global-hl-line-mode 1)	Highlights the current line
(show-paren-mode t)	Toggles emacs' showing matching parentheses
(setq transient-mark-mode t)	Highlights the selected region

Figure 16: Misc. Environment Settings

9 Next Steps

9.1 Learning more about emacs

This Woodnotes guide doesn't even scratch the surface of the power of emacs, but for the writer or author interested in getting down to work, this should be more than enough to allow you to find and use the commands that you need most. There is far, far more about emacs than what's covered here. For example, emacs includes calendar functionality (even the Mayan calendar!), games (even Tetris!), a calculator, a datebook/planner, a fun psychiatrist, and more. If you want to, you can use emacs to read and compose your email, surf newsgroups, and so on. These things may not be immediately useful to writers, but if you decide you like emacs dive in and explore: it's a powerful tool.

Help Menus: If you're interested in learning more about this powerful software package, there are lots of options available to you. Of course, the program has its own help manuals and documentation available on screen. `C-h ?` will give you a list of all help commands and you can choose from there: I'd recommend you start with `C-h t`, "Read the Emacs manual." Scroll down to 'Text' for the parameters of most use to writers. In the GUI version of emacs, simply navigate the Help menu (on consoles, hit `M-`` to do the same thing). For example, as shown in the help menus, the command `C-h k` allows you to find out what command a particular key sequence is bound to, that is "what does `C-x C-o` do?" `C-h b` will show you all keybindings presently in use.

You can find new commands by using emacs' "apropos" utility. Enter `M-x apropos` and hit return. When emacs asks you for an expression, enter what you'd like to know more about, and it will present you a list of all commands that use that expression. For example hit `M-x apropos`, return, and then enter `ispell`. You will see a list of a half a dozen commands, variables, and functions that include the word ispell, including check-ispell-version, checkdoc-ispell, and ispell-change-dictionary. All items shown as

commands are available to you by entering `M-x` and the command name. Functions and variables are not very useful to anyone but emacs lisp programmers, for whom this document is not intended.

On the web: On the web, check out the GNU website: `http://www.gnu.org`, and the emacs wiki, a collaborative website with tips, tricks, and more: `www.emacswiki.org`. Just Googling for emacs will net you a plethora of websites where emacs fans share their tips.

Books: Start with the O'Reilly Guide *Learning GNU Emacs* by Debra Cameron, James Elliott, and Marc Loy, available in the computer section of your favorite bookstore or on line at `www.oreilly.com`. O'Reilly also produces a useful pocket reference guide (also by Debra Cameron) with less explanation but a good list of the most-used commands and settings. Purchase of the books helps support the programmers that work on emacs.

The best way to learn about emacs is simply to start using it. You'll quickly find solutions to your own problems, and the enormous emacs user community is usually more than happy to support you in your effort to learn it. Brace yourself, and come on board. Happy writing!

9.2 Emacs and LaTeX

This Woodnotes guide won't get into the details of LaTeX but let it be said that if you intend to develop documents using TeX or LaTeX software, you could hope for no better text editor than emacs. Find the Auctex package for emacs – it probably came with your distribution anyway – and install it. Auctex will be loaded immediately whenever you open or begin a file with the extension .tex. It makes a whole host of commands available to you at a keystroke and has good documentation that will help you get familiar with its capabilities. It's very convenient to simply hit C-c C-c to run LaTeX on your document and again to view the DVI output file. Auctex also contains provisions to help you organize and create your Bibtex files, and lots more.

10 Acknowledgments

Thanks to Bill Harris, Kai Grossjohann, Chong Yidong (one of the authors of longlines.el), Johann "Myrkraverk" Oskarsson, Jerry Sievers, Marc Girod, Aidan Kehoe, Xah Lee, and everyone else that supplied recommendations and additional resources for this document.

A companion reference card is available for this document at `www.therandymon.com/papers/emacs-writers-cheatsheet.pdf`.