Contents

List of Figures

Aquamacs - An Appreciation

A slightly over-complicated document in LaTeX to explore gotchas in using the lwarp package to prepare documents for automatic re-purposing for the web and e-books.
27-Nov-2020 18:06

1 Is Aquamacs Emacs?

Aquamacs is not a fork of Emacs. It **is** Emacs. What makes it different is ready availability of common macOS keyboard shortcuts, menu conventions and mouse operations. It is easier to get real work done earlier in your GNU Emacs learning phase, which is a life's work.

macOS has waved a magic wand to make this possible. Almost entirely by accident.

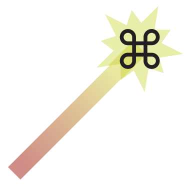

Figure 1: *Izzy Whizzy, Let's Get Bizzy*

Because GNU Emacs is venerable open source, the ⌘ makes no appearance there. This happy disjunction is a gift for Aquamacs beginners. When recalling Emacs commands is too overwhelming for something you can do natively in macOS, just do it the way.

Traditional Emacs knows nothing of the macOS menu bar. Aquamacs Emacs makes great use of it.

Remember that macOS is officially Unix. There are lots of Unix commands that match those of Emacs. Inside the Terminal app many shortcuts display their common unix and Emacs ancestry despite the famous "**G**NU's **N**ot **U**nix" *bon mot* by Richard Stallman in 1983.

A few Emacs commands have found their way into Apple's own text handling frameworks. Ctrl-a, Ctrl-e, Ctrl-b, Ctrl-f, Ctrl-d and Ctrl-t do exactly the same job in Emacs.

1.1 Why bother with Emacs?

You probably use several applications that handle text – Finder, TextEdit, Word, InDesign, Excel, Mail, Terminal, Xcode, LaTeX, Safari, Usenapp …

Most of the biggies in that list handle text in their own private "Integrated Development Environment". Each does many of the same jobs in different ways to the others. Sometimes it is the big hitters sucking you into their eco-systems. Sometimes it is a mistaken goal of Windows compatibility. Other than those, it must be a (S)IDE salad. One has to invest a lot of time learning all those IDE's different ways to do the same basic job.

Does Aquamacs offer a better return on that investment? The answer is a guarded maybe. If you work exclusively in a narrow subject area like vector graphic art, and your boss says it has to be Illustrator, then Emacs is not for you. If you freelance, and switch between Affinity Designer and Adobe Illustrator you might be tempted to look at Aquamacs. If you are authoring books, you might want to flick between Word, Libre Office, InDesign, Affinity

Publisher and TEXshop. Read on! If you are developing software solely in Swift, for the present, stick with Xcode. For any other set of programming languages you should already be using Emacs in one flavour or another. Read on!

For the rest of us, – that potter about with mail, usenet and social media and the odd letter for snail mail, the question can be answered with another question. Why not? Aquamacs is fun, and with more or less every macOS text trick ready and waiting, all you need is Aquamacs, its ever present *scratch* buffer, ⌘C and ⌘V.

Copy or author your text in *scratch*, fix it with the tricks you know or are still learning, and copy paste into the app that wanted it done its way. You have all that Emacs power ready when you need it.

Awesome Emacs Power

Aquamacs **is** Emacs, so it does not stop there. It turns the IDE salad on its head.

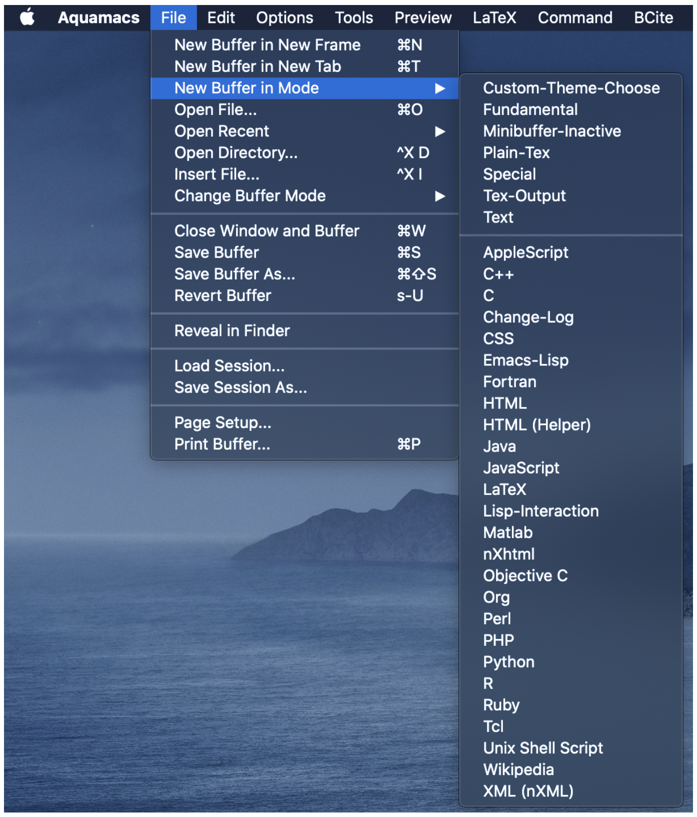

Figure 2: *Mode*

Modes make Emacs an almost universal development environment. If you develop software in more than one language, and/or create documentation in LATEX or for the web, for each of the modes it offers – operations such as completion, indentation, testing, formatting, highlighting – are tailored to the way each ‘language’ expects. The list in the screenshot above shows the common modes. Quite a few can be added.

The screenshot shows a bit of the Mac menu bar in LATEX mode. It was grabbed on the fly as this piece was being written. Underneath those menu items are submenus with their Emacs equivalent key bindings. What an excellent crutch for learning how to keep your fingers on the keyboard! Every mode has its specific keybindings, but everything is as consistent as it possibly can be.

An Analogy

I recently changed cars. I sold the trusty 1998 Audi S8 that had been my daily driver for almost 20 years and bought a newer but still old S8 in its place. This one has a 5.2 liter V10 engine and more electronic toys than anyone needs.

My excuse? I was getting older quicker than my car, and I wanted, not needed, to get a really truly monster before I got too old to enjoy it safely. The mechanic who looks after our cars grinned at me and said “There are no pockets in a shroud”.

I walk up with its keyfob in my pocket; the door opens at first grab. I climb in with my foot on the brake and press start. I press the handbrake off button, select drive and mosey out to the gate, closing the door and buckling in. It then wafts me gently about as a proper dictatormobile should. It is very Macintosh-like. It even has a touch ID that does not work all the time.

But, if I move the gear selector to s, and the air suspension to DYNAMIC all four wheels scrabble for grip with anything more than a gentle stab on the loud pedal, the ride becomes rock hard and I get hugely rewarded for braking late but smoothly and taking the proper line through corners. A complete personality change for the car and the idiot behind the wheel.

This is so like leaving the Mac’s comfort zone for the joy of Emacs. Consider the difference between ⌘F and C-M % which is near the esoteric end of Emacs search commands.

It invokes query-replace-regexp. Let me give you an example of a slightly scary simple one. It asks for two strings of characters. The first is what to look for. The second is how to replace it with something related.

```
["“”]\(\([^"“”]+\)\)["“”]
```

Figure 3: *regexp query*

```
``\1''
```

Figure 4: *regexp replace*

And pray – what does it do? It repairs improperly formed pairs of curly double quote marks. It does so by looking for any double quote mark –it is surprising how many variations there are–, then anything but, until there is another double quote, remembering what was between those two bookends. The replacement is a pair of grave accent characters followed by what was originally between the opening and closing double quotes and then a pair of apostrophe characters. That is the canonical way of setting a double-quoted string in LATEX.

A few remarks are in order.

As anybody who has ever written a computer program can see, there is a glaring bug. What if there were an unpaired double quote, as in 27” iMac? That is something not easy to fix in any language. Do you look right to the end of a long document to determine whether you are still looking for a matching closing quote? Or perhaps some clever algorithm that decides earlier based on what was inside and outside quoted passages? The practical answer lies in the full name of the Emacs command – Interactive query-replace-regexp. It shows you each match and waits for you tell it whether to do it or not with a single character response. One of which is !, meaning do the lot. Emacs has far more clever undo than anything else. It is easy to see when it is matching the wrong way and stop, then undo back to the first mistake. Much quicker than writing a bug-free program in some other language, including Emacs own internal language elisp.

Regexps are rules for searching for patterns. In the real world there are many subtly different regular expression implementations. This is why Emacs for everything is better than dealing with more than one Integrated Development Environment.

The regexp examples above are presented as screengrabbed images. I did this because the LaTeX source is being used to automatically produce web pages and e-book versions. HTML would make a mess of them, and automatic e-book conversion would too.

	Elisp Command	**Function**
⌘N	new-frame-with-new-scratch	Create new buffer
⌘O	find-file-other-frame	Open a file
⌘W	close-window	Close selected window deleting buffer
⌘⇧S	write-file	Save as
⌘A	mark-whole-buffer	Select all text
⌘V	cua-paste (yank)	Paste text
⌘C	clipboard-kill-ring-save	Copy test
⌘⌥X	aquamacs-keyboard-kill-secondary	Cut text from secondary selection
⌘S	save-buffer	Save file
⌘L	goto-line	Go to specified line
⌘F	isearch-forward	Search
⌘G	isearch-repeat-forward	Repeat search
⌘E	aquamacs-use-selection-for find	Use selected text for next search
⌘;	spellcheck-now	Jump to next spelling error
⌘M	iconify-or-deiconify-frame	Minimise window to the dock
⌘.	keyboard-quit	Keyboard quit
⌘,	customize	Show customisation buffer
⌘'	(un)comment-region-or-line	Comment out or in the current line or region if marked
⌘⌫	kill-whole-visual-line	Deletes the current line
⌘⌦	kill-visual-line	Deletes the remainder of the current line
⌘Q	aquamacs-save-buffers-kill-emacs	Save file, exit program
⌘Z	undo	Undo
⌘⇧Z	redo	Redo