Contents

Chapter 1

Installation

I've fetched Emacs from the GitHub mirror repository. The source code resides in ~/developer/software/emacs. Installation instructions reside in the source directory in the file named "INSTALL". I've chosen to use a build directory named "build". From the build directory run configure with:

```
$ ./../configure --with-x-toolkit=gtk3 --with-mailutils \
                 --with-imagemagick --with-xwidgets
```

I'm currently using version 26.1, by running:

```
$ git checkout emacs-26.1
```

There is some issue with highlighting in version 27.050 (potentially font-lock?) that makes it very annoying to write code. In general I should only use stable releases, which are specified on the Emacs website.

Chapter 2

Package Setup

2.1 Gnus

Gnus was a huge pain to setup. I finally got it working with the information at this link.

2.2 Term

Ansi term uses 8 colors for the terminal GUI. We can redefine these colors in an Emacs terminal by customizing their values with `M-x customize-group RET term RET`. The colors must also be configured in `~/.bashrc`.

Chapter 3

Functionality

3.1 Code Tagging and Navigation

Chapter 4

Structure

Chapter 5

Best Practices

This Emacs manual page contains principles to follow when customizing key bindings. Basically, `C-c <letter>` (but not `C-c` followed by another control character) as well as `<f5>` through `<f9>` are free for users to define how they wish.

Chapter 6

To-Do

6.1 Customize Mode Line

I can customize mode-line-format on a per-mode basis (see this StackExchange question). Also, see the Emacs documentation on the subject.

6.2 Disable cursor in pdf-view-mode

I'd like to disable the cursor when viewing a PDF. It's distracting and provides no value. I've tried these two additions to `init.el` to no avail:

```
(add-hook 'pdf-view-mode-hook
          (lambda ()
            (make-variable-buffer-local 'cursor-type)
            (setq cursor-type nil)))
(add-hook 'post-command-hook
          (lambda ()
            (setq cursor-type (if pdf-view-mode t 'nil))))
```

These are the links I've found that address related issues:

How to change the cursor type and color?

Cursor parameters

How can I make the cursor change to block in overwrite mode?

Changing Cursor Dynamically

Cursor Display

Instead of trying to get rid of the cursor, maybe just make its color the same as the background. Check out these links: se and manual.

6.3 Get much better at Gnus

6.4 Flycheck doesn't work with C

I'm getting an error that the flag -std=c++17 doesn't work with C. This error seems somewhat similar.

6.5 load-file from Emacs

This causes cursor color to change and potentially other behavior as well.

6.6 align-current AUCTeX

align-current works well for tables with limited text. However, it propagates line breaks and so takes up a needless number of lines for long text sections that have to be manually adjusted. Also, I'd like to be able to run this globally for a whole buffer. The most similar action is align-entire but this aligns all tables to one another; I'd like aligning to be done on a per-table basis.

6.7 Consider switching to sane-term from multi-term

sane-term seems like it might be a better alternative to multi-term.

6.8 Single quote delimeter not working

Pressing ́twice causes three single quotes to be inserted.

6.9 Command that closes all buffers for which underlying file was deleted

Probably better, just do this automatically. One worry is if file was accidentally deleted and buffer allows you to keep the file alive. Still, existing buffers shouldn't be used as a sort of backup.

6.10 Magit bug missing headers

I'm getting the error: "BUG: missing headers nil" when using magit with ghub. This is a known issue and is tracked here. I guess the solution is to just wait for someone to fix it there.

6.11 Verilog mode escape

When in evil insert mode and the cursor position is after a reg or wire, pressing escape tries to vectorize the reg/wire instead of entering evil normal state.

6.12 Verilog code tagging

Verilog mode should have code tagging and navigation. See this link for ideas on code tagging.

6.13 Better Code Tagging

The problem with RTags is that it only works for C-based languages. gtags should work for many others. See this link. It probably makes sense to use the ggtags interface. Also use the helm interface. This may also be the wrong question, but maybe it's also possible to use rtags as a backend for a common gtags frontend?

6.14 Create Add-On that allows you to edit Anki HTML in Emacs

Adapt this for your needs. That add-on should contain nearly all of the required functionality.

6.15 Get XWidget Webkit working

See this.

6.16 Get LaTeX SVGs working in EWW

EWW is unable to render latex/mathjax directly and so renders the resulting svg instead. This appears as dark text on a dark background. See this post.

6.17 Improve org-books

1. Query user for file when running org-capture and base title on that.
2. Make table of contents entries links to the corresponding page in the pdf.
3. Write a program to extract a table of contents from pdf. Run this automatically when running org-capture.
4. Bind org-capture to a keybinding.
5. Write scripts to get info from Goodreads and Amazon (Goodreads API here).
6. Add description property so that you can make comments on the quality (e.g. "Definitive book on GR").
7. Potentially use Wikipedia categories to categorize books in list and in directories.

6.18 Org Capture Extension

Consider using org-capture-extension which allows you to run org-capture from a browser.

6.19 Git Bug

Consider using git-bug instead of this file to track issues and improvements.

6.20 Eyebrowse

Use eyebrowse to easily navigate between Emacs workstations.

6.21 Setup BBDB or other contact list for GNUs

bbdb keeps a contact list that can be used in gnus. Set this up, or look for an alternative (potentially one that does not require manual additions). Note that there is also a company backend for this, company-bbdb.

6.22 Company specifies backend for each completion

It would be nice to be able to show backend used for each completion with company. This could be a configurable setting in company, for instance. Maybe look at company-box for a possible way to implement this.

6.23 Implement method to see what is causing lag in Emacs

It would be really nice to be able to see what is causing Emacs to lag in certain contexts. This could be enabled by calling a sort of record function and then disabling it when done. When the record function is toggled off, display a list of functions called and how much collective time they took, ordered by that time.

6.24 Create a new gdb many windows mode that spans 2 screens

This should also have a function to restore windows. It should delegate most responsibility to GDB.

6.25 Change company completion face

The red text face used for company completion is ugly. Find a thematic alternative.

6.26 Fix YCMD no flags error

YCMD sometimes complains and repeatedly throws a flags missing error. Figure out what causes this (look at the YCMD code), then figure out how to solve it. This probably involves reconfiguration the ycm extra conf file.

6.27 Keep company-box?

Consider the functionality of company-box and potentially get rid of it.

6.28 Completion drop-down vs fill-in

Sometimes completions provide a drop-down and sometimes it doesn't and instead presents a sole completion as a differently colored rest of the word. What logic regulates this? Is it when only a single completion is available? Is this disadvantageous for a sole completion when the dropdown could also provide documentation for the completion?

6.29 Improve GDB behavior for opening windows

The current GDB split window behavior isn't great. It splits up existing windows into overly small sizes. I'm not exactly sure what I want here, but I'd like a strong priority given to the gdb buffer and the source buffer. In any event these two links should be helpful: this and this.

6.30 Enable ability in term mode when at command line to insert at point instead of going to end

This would allow me to navigate in normal mode and then insert like normal.

```
(evil-collection-define-key 'normal 'term-mode-map (kbd "i")
  (lambda ()
    (interactive)
    (if (eq (line-number-at-pos)
            (+ (evil-count-lines (point-min) (point-max)) 1))
        (let* ((col-str (what-cursor-position))
               (col (string-to-number
                     (when (string-match ".*column=" col-str)
                       (replace-match "" nil nil col-str)))))
          (progn (evil-insert-state)
                 (term-send-end)
                 (term-show-maximum-output)
                 (let* ((col-new-str (what-cursor-position))
                        (col-new (string-to-number
                                  (when (string-match ".*column=" col-new-str)
                                    (replace-match "" nil nil col-new-str)))))
                   (while (> col-new col)
                     (progn (execute-kbd-macro (kbd "C-b"))
                            (message (format "%d %d" col col-new))
                            (setq col-new (1- col-new)))))))
      (progn (evil-insert-state)
             (term-show-maximum-output)))))
```

This doesn't work because edits made to the buffer line in term line mode are not affected in term-char mode. This would take a lot more work.

6.31 Prevent sudo directories from staying in recentf list

We can probably configure this with tramp, see manual.

6.32 Store source URL of book and write function to detect if changes have occurred

Use a hash function for this. Hopefully there's a way to do this without redownloading all files each time.

6.33 Get helm-make working

See here.

6.34 Code completion should ignore case

Setup company in code completion modes to ignore entered case and fill in the correct case regardless.

6.35 Setup rr debugger with gdb interface

See this link on setting it up. See this other link on what it can do.

6.36 Function to determine an installed packages dependencies

Some package is using magit-popup which is causing a warning each time magit is used. Figure out which package. More generally, write a function to figure out which packages use a given dependency. Additionally, have another function to list all dependencies for a given input.

6.37 Fix info manual building with straight

Most info manuals aren't built.

6.38 Make c-style-alist style options buffer local from clang-format

c-style-alist specifies a number of style options that should be set in the same way as the other style options from clang-format.

6.39 Setup smartparens

Use smartparens instead of other packages.

6.40 flycheck clang tidy automatically find compile commands

Find compile_commands.json recursively in current project directory, rather than needing to specify it.