Refactoring Dotfiles For Colemak

Rohit Goswami · 2020-05-02 · workflow · dots

> *A more actionable follow up to my personal recollections relating to my switch to Colemak.*

Background

I have, in the past written about how I made the switch to Colemak. However, until recently, I was still trying to mimic the VIM keybindings from QWERTY. This is a post where I discuss the changes I made to ensure that I never have to stretch my fingers in odd ways again. The main idea is expressed well by vim-colemak.

```
Colemak layout:                 |                QWERTY layout:
`12345 67890-=     Move around: |  (instead of)  `12345 67890-=
 qwfpg jluy;[]\         e       |       k         qwert yuiop[]\
 arstd HNEIo'         h   i     |     h   l       asdfg HJKL;'
 zxcvb km,./            n       |       j         zxcvb nm,./
```

Sudoers

The `sudo` command does not automatically pick up on your keyboard layout. It is best to set this explicitly. Use `visudo` and un-comment `Defaults env_keep += "LANG LANGUAGE LINGUAS LC_* _XKB_CHARSET"`, or:

```
su
echo 'Defaults env_keep += "LANG LANGUAGE LINGUAS LC_* _XKB_CHARSET"' >> /etc/sudoers
```

Emacs

Though I have mentioned publicly, that I was using the regular QWERTY motion keys, I realized I had actually started to use the mouse more often, simply because it was a pain to move around. Thankfully, `emacs` has evil-colemak-basics, which is fabulous. For reference, these make it really easy for QWERTY users to make the switch if they're previously used to VIM bindings.

Colemak	Qwerty	Action	States At Qwerty position?		Remarks
`h`, `n`, `e`, `i`	`h`, `j`, `k`, `l`	move	`mnvo`	yes	
`k`, `K`	`n`, `N`	search next/previous	`mnvo`	yes	
`u`, `U`	`i`, `I`	insert	`_nv_`	yes	
`l`	`u`	undo	`_nv_`	yes	
`N`	`J`	join lines	`_nv_`	yes	
`E`	`K`	lookup	`mnv_`	yes	
`u`	`i`	inner text object keymap	`___o`	yes	
`f`, `F`	`e`, `E`	jump to end of word	`mnvo`	yes	with `t-f-j` rotation
`t`, `T`	`f`, `f`	jump to character	`mnvo`	yes	with `t-f-j` rotation
`j`, `J`	`t`, `T`	jump until character	`mnvo`	no	with `t-f-j` rotation

Colemak	Qwerty	Action	States At Qwerty position?	Remarks
`j`, `J`	`e`, `E`	jump to end of word	`mnvo` no	without `t-f-j` rotation

Where the table above is from the fantastic readme.

I still had some issues, mostly relating to searching in buffers, so I ended using `swiper-isearch` more which is a bonus too.

Visual Lines

Since I tend to keep `visual-line-mode` all the time, it makes sense to actually swap working with lines and visual lines. To work this through this needs evil-better-visual-line.

```
(use-package! evil-better-visual-line
  :after evil-colemak-basics
  :config
  (evil-better-visual-line-on)
  (map! :map evil-colemak-basics-keymap
        (:nvm "n" 'evil-better-visual-line-next-line
         :nvm "e" 'evil-better-visual-line-previous-line
         :nvm "g n" 'evil-next-line
         :nvm "g e" 'evil-previous-line))
)
```

Pdf-Tools

For my `doom-emacs` configuration, I also set the following map:

```
(after! pdf-view
  (add-hook! 'pdf-view-mode-hook (evil-colemak-basics-mode -1))
 (map!
   :map pdf-view-mode-map
   :n "g g"          #'pdf-view-first-page
   :n "G"            #'pdf-view-last-page
   :n "N"            #'pdf-view-next-page-command
   :n "E"            #'pdf-view-previous-page-command
   :n "e"            #'evil-collection-pdf-view-previous-line-or-previous-page
   :n "n"            #'evil-collection-pdf-view-next-line-or-next-page
 )
```

Where the most important thing is the hook which removes the `evil-colemak-basics` binding. Since it is a single mode and hook, `after-hook!` is the same as `after-hook` [^fn:1].

Window Management

Somehow these are not part of the `evil-colemak` defaults.

```
(after! evil
  (map! :map evil-window-map
        (:leader
         (:prefix ("w" . "Select Window")
          :n :desc "Left"  "h" 'evil-window-left
          :n :desc "Up"    "e" 'evil-window-up
          :n :desc "Down"  "n" 'evil-window-down
          :n :desc "Right" "i" 'evil-window-right
          ))
        ))
```

Search

Harmonizing with Vimium.

```
(after! evil (map! :map evil-motion-state-map
                   (:n :desc "Previous match" "K" 'evil-ex-search-previous
                    :n :desc "Next match" "k" 'evil-ex-search-next
                    :n :desc "Forward search" "/" 'evil-search-forward
                    )
                   ))
```

Page Movement

Though this is more of a personal preference, I find it more natural to bind N and E to page-wise movement instead of join lines and lookup, since I almost never use those commands, and the movement keys echo what I expect elsewhere.

```
(after! evil
  (map! :map evil-colemak-basics-keymap
      :nv "N" 'evil-scroll-page-up
      :nv "E" 'evil-scroll-page-down)
  )
```

Evil Org

Annoyingly, `evil-org-mode` had a map which kept overriding all my other settings. Thankfully it has a helper variable to set movement. I also do not need this anyway, at-least not by default.

```
(after! org
  (remove-hook 'org-mode-hook 'evil-org-mode)
  (setq evil-org-movement-bindings
        '((up . "e") (down . "n")
          (left . "h") (right . "i"))
        )
)
```

Vimium

I use the excellent vimium to make Chrome be a little less annoying. Luckily the Wiki seems to have a reasonable suggestion for colemak. The basic idea is to migrate the underlying keys directly to ensure very few manual changes are required.

```
mapkey n j
mapkey N J
mapkey e k
mapkey E K
mapkey i l
mapkey I L
mapkey k n
mapkey K N
mapkey l i
mapkey L I
mapkey j e
mapkey J E
```

Tridactyl

I still use the fantastic tridactyl for Firefox when I can. However, the bindings are slightly more involved, since there is no equivalent for the `mapkey` which `Vimium` has.

```
" Rebinds for colemak
" hjkl --> hnei
bind h scrollpx -50
bind n scrollline 10
bind e scrollline -10
bind i scrollpx 50
" HJKL --> HNEI
bind H back
bind N tabprev
bind E tabnext
bind I forward
```

Vim

For a lot of terminal edits, `vim` is still my editor of choice, and vim-colemak works without any trouble in my configuration.

Zsh

To ensure uniform bindings, I used to use `bindkey -v` but will need some minor changes to that set up. I based this part of my configuration off the bindings of bunnyfly.

```
bindkey -v
# Colemak.
  bindkey -M vicmd "h" backward-char
  bindkey -M vicmd "n" down-line-or-history
  bindkey -M vicmd "e" up-line-or-history
  bindkey -M vicmd "i" forward-char
  bindkey -M vicmd "s" vi-insert
  bindkey -M vicmd "S" vi-insert-bol
  bindkey -M vicmd "k" vi-repeat-search
  bindkey -M vicmd "K" vi-rev-repeat-search
  bindkey -M vicmd "l" beginning-of-line
  bindkey -M vicmd "L" end-of-line
  bindkey -M vicmd "j" vi-forward-word-end
  bindkey -M vicmd "J" vi-forward-blank-word-end

# Sane Undo, Redo, Backspace, Delete.
  bindkey -M vicmd "u" undo
  bindkey -M vicmd "U" redo
  bindkey -M vicmd "^?" backward-delete-char
  bindkey -M vicmd "^[[3~" delete-char

# Keep ctrl+r searching
  bindkey -M viins '^R' history-incremental-pattern-search-forward
  bindkey -M viins '^r' history-incremental-pattern-search-backward
```

Zathura

There is no better `pdf` viewer than zathura, and it also works for `djvu` and friends. As a plus point, it normally has very reasonable `vim` bindings, and an excellent configuration system, so we will leverage that. The best part is that we can just add to it using `include zathuraColemak` or whatever so as to be minimally invasive.

```
map h scroll left
map n scroll down
map e scroll up
map i scroll right

map N scroll half-down
map E scroll half-up

map k search forward
map K search backward

# For TOC navigation
map [index] o toggle_index

# hjkl →  hnei
map [index] n navigate_index down
map [index] e navigate_index up
map [index] h navigate_index collapse
map [index] i navigate_index expand

map [index] H navigate_index collapse-all
map [index] I navigate_index expand-all
```

Zathura is a complicated beast, however, and my full configuration contains a lot more information.

i3

I have some bindings set up in terms of $left$right $up$and$down, so it was simple to re-bind them.

```
set $left h
set $down n
set $up e
set $right i
```

MailMate

Sadly, one of the email clients I do use regularly of late is MailMate. It supports a rather rich set of keybindings placed, e.g. with `"~/Library/Application\ Support/MailMate/Resources/KeyBindings/"` configured as follows:

```
{
        "c"       = "newMessage:";
        "/"       = "searchAllMessages:";
        "n"       = "nextMessage:";
        "e"       = "previousMessage:";
        "h" = "collapseThread:";
        "i" = "expandThread:";
        "H" = "rootOfThread:";
        "I" = "lastOfThread:";
        "N" = "nextThread:";
        "E" = "previousThread:";
        "o"       = "openMessages:";
        "x" = ( "deleteMessage:", "nextMessage:" ); // Defaults to going to the previous message
        "a"       = "archive:";
        "s"       = "toggleFlag:";
        "!"       = "moveToJunk:";
        "r"       = "reply:";
        "R"       = "replyAll:";
        "f"       = "forwardMessage:";
        "^s"      = "saveDocument:";
        "u" = "toggleReadState:";
}
```

Conclusions

That seems to be it for now. If I think of more programs I use regularly which allow VIM bindings, or keybindings in general, I'll probably just update this post. My full dotfiles are present here, and now include a `colemak` target.
