learning-gun-emacs

Hack Chyson

September 5, 2019

Contents

1 DONE Customizing Emacs **1**
- 1.1 Using Custom . . . 2
- 1.2 Configuration Load Order When Start Up . . . 2
- 1.3 Customizing Your Key Bindings . . . 2
- 1.4 Setting Emacs Variables . . . 4
- 1.5 Starting Modes via Auto-Mode Customization . . . 5
- 1.6 Functions . . . 5
- 1.7 Making word abbreviations part of your startup . . . 5

1 DONE Customizing Emacs

three ways to customize Emacs:

1. Options
2. Custom
3. Lisp code

No matter what method you use, though, the .emacs startup file is modified. Custom modifies it for you when you save settings through that interface. The Options menu invokes Custom behind the scenes; when you choose Save Options, Custom again modifies .emacs.

1.1 Using Custom

Bounding	Function	Description
	customize	

1.2 Configuration Load Order When Start Up

Loading order:

1. site-start.el
2. configuration
   (a) .emacs.elc (compiled version of .emacs.el)
   (b) .emacs.el
   (c) .emacs
3. default.el

1.3 Customizing Your Key Bindings

Term	Description
keymap	a collection of key bindings
global-map	the most basic default key bindings
local-map	specific to a single buffer
ctl-x-map	keymap for C-x
esc-map	keymap for Esc

When you type a key, Emacs first looks it up in the current buffer's local map (if any). If it doesn't find an entry there, it looks in global-map.

What happens with commands that are bound to multiple keystrokes? The answer is that the keys C-x , Esc , and C-c are actually bound to special internal functions that cause Emacs to wait for another key to be pressed

and then to look up that key's binding in another map.

Caution: You can use Meta in place of Esc , but the bindings are still stored in the esc-map .

Three ways to define your own key bindings:

```
(define-key keymap "keystroke" 'command-name)
(global-set-key "keystroke" 'command-name)
(local-set-key "keystroke" 'command-name)
```

Special character conventions:

```
| Special Character | Definition                  |
|-------------------+-----------------------------|
| \C-x              | C-x (where x is any letter) |
| \C-[ or \e        | Esc                         |
| \M                | Meta                        |
| \C-j or \n        | Newline                     |
| \C-m or \r        | Enter                       |
| \C-i or \t        | Tab                         |
```

Note that control characters are case-insensitive — that is, -A is the same thing as -a. However, the characters that follow control characters may be case-sensitive; -ae could be different from -aE .

The function define-key is the most general because it can be used to bind keys in any keymap.

```
(global-set-key "\C-xl" 'goto-line)
equal to
(define-key global-map "\C-xl" 'goto-line)
(define-key ctl-x-map "l" 'goto-line)
```

Two ways to make the change in your .emacs to take effect:

Bounding	Function	Description
	eval-buffer	
C-x C-e	eval-last-sexp	

To unset key bindings:

```
(global-unset key "\C-xl")
(define-key ctl-x-map "l" nil)
```

1.4 Setting Emacs Variables

To set the value of a variable, use the setq function in your .emacs.

```
(setq variable-name variable-value)
```

Several Emacs variables can have different values for each buffer (local values) as well as a default value. Such variables assume their default values in buffers where the local values are not specified.

When you set the value of a variable with setq , you are actually setting the local value. The way to set default values is to use setq-default instead of setq.

```
(setq-default variable-name variable-value)
```

Unfortunately, there is no general way to tell whether a variable has just one global value or hasdefault and local values (except, of course, by looking at the Lisp code for the mode). Therefore the best strategy is to use a plain setq , unless you find from experience that a particular variable doesn't seem to take on the value you setq it to — in which case you should use setq-default .

Variable	Description	Default
kill-ring-max		60
auto-save-interval		300
case-fold-search		t
shell-file-name	decide which shell to start	/bin/bash
dired-garbage-files-regexp		
tab-width		4
auto-mode-alist		
major-mode		
inhibit-default-init		nil
compile-command		make -k
compilation-error-regexp-alist		

1.5 Starting Modes via Auto-Mode Customization

The assocations of suffix and major mode are contained in a variable auto-mode-alist .
auto-mode-alist is a list of pairs (regexp . mode), where regexp is a regular expression and mode is the name of a function that invokes a major mode.

Syntax:

```
(setq auto-mode-alist (cons '("<suffix>" . <major-mode>) auto-mode-alist))
```

1.6 Functions

Bounding	Function	Description
edit-tab-stops		

1.7 Making word abbreviations part of your startup

To define word abbreviation and make them part of your startup, add these lines to your .emacs file:

```
(setq-default abbrev-mode t)
(read-abbrev-file "~/.emacs.d/.abbrev_defs")
(setq save-abbrevs t)
```