Emacs

David Florness

October 3rd, 2019
We spend a lot of time editing text

writing programs
writing essays
taking notes
writing email

Why not create a program to make typing text as easy as possible?

Emacs is one such attempt at a program.

Let me show you why it's the best attempt thus far.

David Florness
Emacs
October 3rd, 2019
We spend a lot of time editing text

1. writing programs

Emacs is one such attempt at a program. Let me show you why it's the best attempt thus far.

David Florness
Emacs
October 3rd, 2019
We spend a lot of time editing text

1. writing programs
2. writing essays
We spend a lot of time editing text

1. writing programs
2. writing essays
3. taking notes
We spend a lot of time editing text

1. writing programs
2. writing essays
3. taking notes
4. writing email
We spend a lot of time editing text

1. writing programs
2. writing essays
3. taking notes
4. writing email

Why not create a program to make typing text as easy as possible?
We spend a lot of time editing text

1. writing programs
2. writing essays
3. taking notes
4. writing email

Why not create a program to make typing text as easy as possible?

Emacs is one such attempt at a program.
We spend a lot of time editing text

1. writing programs
2. writing essays
3. taking notes
4. writing email

Why not create a program to make typing text as easy as possible?

Emacs is one such attempt at a program.

Let me show you why it’s the best attempt thus far.
What is Emacs?

A "display editor" designed and written in the 1970's by Richard Stallman, who was very impressed with the Stanford AI Lab's E editor and wanted to add macros to it. The name "Emacs" means Editing Macros. Officially, Emacs is "the extensible, customizable, self-documenting, real-time display editor." (in shell: info emacs)
What is Emacs?

A “display editor” designed and written in the 1970’s by Richard Stallman, who was very impressed with the Stanford AI Lab’s E editor and wanted to add macros to it. The name “Emacs” means Editing Macros.
What is Emacs?

A “display editor” designed and written in the 1970’s by Richard Stallman, who was very impressed with the Stanford AI Lab’s E editor and wanted to add macros to it. The name “Emacs” means Editing Macros.

Officially, Emacs is “the extensible, customizable, self-documenting, real-time display editor.” (in shell: info emacs)
By a display editor we mean an editor in which the text being edited is normally visible on the screen and is updated automatically as the user types his commands. No explicit commands to 'print' text are needed.
By a display editor we mean an editor in which the text being edited is normally visible on the screen and is updated automatically as the user types his commands. No explicit commands to ‘print’ text are needed.
Customizable?

Everything in Emacs is either a LISP variable waiting to be changed (C-h v) or a LISP function waiting to be invoked (C-h c). All keys you type simply call LISP functions that perform operations on the text. You may think of Emacs as a LISP machine trapped inside of a program.
Customizable?

1. Everything in Emacs is either a

   a LISP variable waiting to be changed (C-h v)
   b LISP function waiting to be invoked (C-h c)
   c all keys you type simply call LISP functions that perform operations on
       the text

   you may think of Emacs as a LISP machine trapped inside of a
   program
Customizable?

1. Everything in Emacs is either a
   - LISP variable waiting to be changed (`C-h v`) everything you see on the screen can be modified.
Everything in Emacs is either a

- LISP variable waiting to be changed (C-h v)
  everything you see on the screen can be modified
- LISP function waiting to be invoked (C-h c)
  all keys you type simply call LISP functions that perform operations on
  the text
Customizable?

1. Everything in Emacs is either a
   - LISP variable waiting to be changed (C-h v)
     everything you see on the screen can be modified
   - LISP function waiting to be invoked (C-h c)
     all keys you type simply call LISP functions that perform operations on the text

2. you may think of Emacs as a LISP machine trapped inside of a program
While Emacs is running, you can tell Emacs to load new bits of LISP code for your use. While the core of Emacs is written in C, most of it is bits of LISP code that define how the editor behaves. Emacs can install packages, which are nothing more than tarballs of source-code files containing LISP function & variable declarations.
While Emacs is running, you can tell Emacs to load new bits of LISP code for your use.
While Emacs is running, you can tell Emacs to load new bits of LISP code for your use.

While the core of Emacs is written in C, most of it is bits of LISP code that define how the editor behaves.
While Emacs is running, you can tell Emacs to load new bits of LISP code for your use.

While the core of Emacs is written in C, most of it is bits of LISP code that define how the editor behaves.

Emacs can install packages.

Packages are nothing more than tarballs of source-code files containing LISP function & variable declarations.
Most keys simply insert themselves into the buffer. Others (usually starting with CTRL or ALT) call LISP functions. If you want to call a LISP function that has no associated key, do \texttt{M-x} (i.e. ALT-x).
Most keys simply insert themself into the buffer
Most keys simply insert themself into the buffer

Others (usually starting with CTRL or ALT) call LISP functions
Interacting with Emacs

Most keys simply insert themselves into the buffer.

Others (usually starting with CTRL or ALT) call LISP functions.

If you want to call a LISP function that has no associated key, do `M-x` (i.e. ALT-x).
A word about keybindings

Vim's keybindings are better. Anyone who says otherwise is lying.

So install Evil Mode, the Extensible VI Layer for Emacs:

```lisp
(package-install 'evil)
(evil-mode)
```

Having said that, the Emacs bindings are worth knowing. They're the default in most shells and in GNU readline, which is used by just about every REPL.
Vim’s keybindings are better.
Vim’s keybindings are better. Anyone who says otherwise is lying.
A word about keybindings

Vim’s keybindings are better. Anyone who says otherwise is lying.

So install Evil Mode, the Extensible VI Layer for Emacs:

```lisp
(package-install 'evil)
(evil-mode)
```
A word about keybindings

Vim’s keybindings are better. Anyone who says otherwise is lying.

So install Evil Mode, the Extensible VI Layer for Emacs:

```lisp
(package-install 'evil)
(evil-mode)
```

Having said that, the Emacs bindings are worth knowing. They’re the default in most shells and in GNU readline, which is used by just about every REPL.
A word about keybindings

Vim’s keybindings are better. Anyone who says otherwise is lying.

So install Evil Mode, the **Extensible VI Layer** for Emacs:

```
(package-install 'evil)
(evil-mode)
```

Having said that, the Emacs bindings are worth knowing. They’re the default in most shells and in GNU readline, which is used by just about every REPL.

The bindings themself were created for the space cadet keyboard
The legendary Emacs packages

Let’s take a look at some of the greatest Emacs packages.
The legendary Emacs packages

Let’s take a look at some of the greatest Emacs packages.

1. *Dired (builtin)* — easily traverse filesystem and perform file operations
The legendary Emacs packages

Let’s take a look at some of the greatest Emacs packages.

1. Dired (builtin) — easily traverse filesystem and perform file operations
2. Ivy/Helm — turn \texttt{M-x} into a fuzzy finder
The legendary Emacs packages

Let’s take a look at some of the greatest Emacs packages.

1. Dired (builtin) — easily traverse filesystem and perform file operations
2. Ivy/Helm — turn M-x into a fuzzy finder
3. Magit — the greatest interface to Git
The legendary Emacs packages

Let’s take a look at some of the greatest Emacs packages.

1. Dired (builtin) — easily traverse filesystem and perform file operations
2. Ivy/Helm — turn `M-x` into a fuzzy finder
3. Magit — the greatest interface to Git
4. AUCTeX — features to make typing LaTeX easier
The legendary Emacs packages

Let’s take a look at some of the greatest Emacs packages.

1. Dired (builtin) — easily traverse filesystem and perform file operations
2. Ivy/Helm — turn M-x into a fuzzy finder
3. Magit — the greatest interface to Git
4. AUCTeX — features to make typing La,TeX easier
5. SLIME/CIDER/Geiser — interactive programming for LISPs
The legendary Emacs packages

Let’s take a look at some of the greatest Emacs packages.

1. Dired (builtin) — easily traverse filesystem and perform file operations
2. Ivy/Helm — turn M-x into a fuzzy finder
3. Magit — the greatest interface to Git
4. AUCTeX — features to make typing LaTeX easier
5. SLIME/CIDER/Geiser — interactive programming for LISP languages
6. Yasnippet — snippets

Language server protocol client via lsp-mode or eglot gives you IDE-like editing features for any programming language that has an LSP server (most do)
Let’s take a look at some of the greatest Emacs packages.

1. Dired (builtin) — easily traverse filesystem and perform file operations
2. Ivy/Helm — turn \texttt{M-x} into a fuzzy finder
3. Magit — the greatest interface to Git
4. AUCTeX — features to make typing La\TeX{} easier
5. SLIME/CIDER/Geiser — interactive programming for LISP
6. Yasnippet — snippets
7. Language server protocol client via \texttt{lsp-mode} or \texttt{eglot}
   - gives you IDE-like editing features for any programming language that has an LSP server (most do)
Let’s take a look at some of the greatest Emacs packages.

1. **Dired (builtin)** — easily traverse filesystem and perform file operations
2. **Ivy/Helm** — turn `M-x` into a fuzzy finder
3. **Magit** — the greatest interface to Git
4. **AUCTeX** — features to make typing LaTeX easier
5. **SLIME/CIDER/Geiser** — interactive programming for LISPs
6. **Yasnippet** — snippets
7. **Language server protocol client** via `lsp-mode` or `eglot`
   - gives you IDE-like editing features for any programming language that has an LSP server (most do)
8. **mu4e / gnus** — read email inside Emacs
The legendary Emacs packages

Let’s take a look at some of the greatest Emacs packages.

1. Dired (builtin) — easily traverse filesystem and perform file operations
2. Ivy/Helm — turn M-x into a fuzzy finder
3. Magit — the greatest interface to Git
4. AUCTeX — features to make typing La,TeX easier
5. SLIME/CIDER/Geiser — interactive programming for LISPs
6. Yasnippet — snippets
7. Language server protocol client via lsp-mode or eglot
   - gives you IDE-like editing features for any programming language that has an LSP server (most do)
8. mu4e / gnus — read email inside Emacs
9. Comint mode (builtin) — your buffer is a REPL; makes interactive programming a breeze
Whoa there!

Emacs vs UNIX

David Florness

Emacs

October 3rd, 2019
Whoa there! Emacs is a blatant violator of the UNIX philosophy: every program should do one thing, well!
Whoa there! Emacs is a blatant violator of the UNIX philosophy: every program should do one thing, well!

Every Vim user ever:
Whoa there! Emacs is a blatant violator of the UNIX philosophy: every program should do one thing, well!

Every Vim user ever:

*Emacs is a great operating system, lacking only a decent editor.*
Emacs loves UNIX

Emacs does not try to replace those beloved UNIX commands, but rather gives a clean, consistent, interactive way to use them.
Emacs loves UNIX

Emacs does not try to replace those beloved UNIX commands, but rather gives a clean, consistent, interactive way to use them.

M-x grep
Emacs loves UNIX

Emacs does not try to replace those beloved UNIX commands, but rather gives a clean, consistent, interactive way to use them.

M-x grep

In the Magit status buffer, type $ to see that Magit is using Git directly
Emacs loves UNIX

Emacs does not try to replace those beloved UNIX commands, but rather gives a clean, consistent, interactive way to use them.

M-x grep

In the Magit status buffer, type $ to see that Magit is using Git directly.

Dired is just an interactive cd and ls.
On the subject of startup times

1. Vim takes way less time to startup

But so what? Why in the world are you rebooting your editor every time you want to edit a file?

Emacs can be run in daemon mode:

```
M-x server-start
```

Then, you can connect to the Emacs server with the Emacs client:

```
$ emacsclient file
```
On the subject of startup times

1. Vim takes way less time to startup
2. But so what?

Emacs can be run in daemon mode:

M-x server-start

Then, you can connect to the Emacs server with the Emacs client:

$ emacsclient file
On the subject of startup times

1. Vim takes way less time to startup
2. But so what? Why in the world are you rebooting your editor every time you want to edit a file?
On the subject of startup times

1. Vim takes way less time to startup
2. But so what? Why in the world are you rebooting your editor every time you want to edit a file?
3. Emacs can be run in daemon mode:
   M-x server-start
4. Then, you can connect to the Emacs server with the Emacs client:
   $ emacsclient file
So you want to give it a shot... I

There is a builtin tutorial
Emacs tutorial. See end for copying conditions.

Emacs commands generally involve the CONTROL key (sometimes labeled CTRL or CTX) or the META key (sometimes labeled EDIT or ALT). Rather than write that in full each time, we'll use the following abbreviations:

C-<chr> means hold the CONTROL key while typing the character <chr>
  Thus, C-f would be: hold the CONTROL key and type f.
M-<chr> means hold the META or EDIT or ALT key down while typing <chr>.
  If there is no META, EDIT or ALT key, instead press and release the
  ESC key and then type <chr>. We write <ESC> for the ESC key.

Important note: to end the Emacs session, type C-x C-c. (Two characters.)
To quit a partially entered command, type C-g.
To stop the tutorial, type C-x k, then <Return> at the prompt.
The characters ">>" at the left margin indicate directions for you to
try using a command. For instance:

[Middle of page left blank for didactic purposes. Text continues below]

>> Now type C-v (View next screen) to scroll down in the tutorial.
   (go ahead, do it by holding down the CONTROL key while typing v).
   From now on, please do this whenever you reach the end of the screen.
Spacemacs

So you want to give it a shot... III

A community-driven Emacs distribution
The best editor is neither Emacs nor Vim, it's Emacs and Vim!
1 Description

This layer adds support for the Python language.

1.1 Features

- Auto-completion using anaconda-mode
- Code Navigation using anaconda-mode
- Documentation Lookup using anaconda-mode and pylookup
- Test Runners using nose or pytest
- Virtual Environment using pyenv or pyenv
- semantic mode is enabled
- PEP8 compliant formatting via YAPF