EMACS, THE ULTIMATE

Trung Ta

Oracle Labs Australia & National University of Singapore

Brisbane, Australia – August 9th, 2017
Which editor do you use the most?
CASE STUDY: REAL PROGRAMMERS
CASE STUDY: REAL PROGRAMMERS

nano? REAL PROGRAMMERS USE emacs

HEY. REAL PROGRAMMERS USE vim.

WELL, REAL PROGRAMMERS USE ed.

NO, REAL PROGRAMMERS USE cat.

REAL PROGRAMMERS USE A MAGNETIZED NEEDLE AND A STEADY HAND.

EXCUSE ME, BUT REAL PROGRAMMERS USE BUTTERFLIES.

THEY OPEN THEIR HANDS AND LET THE DELICATE WINGS FLAP ONCE.

THE DISTURBANCE RIPPLES OUTWARD, CHANGING THE FLOW OF THE EDDY CURRENTS IN THE UPPER ATMOSPHERE.

WHICH ACT AS LENSES THAT DEFLECT INCOMING COSMIC RAYS, FOCUSING THEM TO STRIKE THE DRIVE PLATTER AND FLIP THE DESIRED BIT.

NICE. 'COUSE, THERE'S AN EMACS COMMAND TO DO THAT.

OH YEAH! GOOD OL' C-x M-c M-butterfly...

DAMMIT, EMACS.

https://xkcd.com/378/
M-x butterfly
Outline

1. Emacs, the basic
2. Emacs, the advance
3. Emacs, the ultimate
4. Getting started
5. Conclusion
A few notes

- The talk title is borrowed from: lambda-the-ultimate.org
- Introduce what I know that Emacs can do
- Many things that I don’t know or haven’t got chances to use
- No intention to compare Emacs with other editors/IDEs
A few notes

- The talk title is borrowed from: lambda-the-ultimate.org
- Introduce what I know that Emacs can do
- Many things that I don’t know or haven’t got chances to use
- No intention to compare Emacs with other editors/IDEs
- The first time I give an Emacs talk 😊
Emacs, the basic
How do we interact with editor?
How do we interact with editor?
How do we interact with editor?
How can we edit text unproductively?

- Frequently moving hand between keyboard & mouse
- Frequently moving palm around keyboard keys
- Not using convenient editing features

▶ Use ←, →, ↑, ↓ keys instead of Home, End, PageUp, PageDown
▶ Use an editor with few convenient editing features

⇒ Emacs helps to avoid these problems!
How can we edit text unproductively?

- Frequently moving hand between keyboard & mouse

⇒ Emacs helps to avoid these problems!
How can we edit text unproductively?

- Frequently moving hand between keyboard & mouse
- Frequently moving palm around keyboard keys

Emacs helps to avoid these problems!
How can we edit text unproductively?

- Frequently moving hand between keyboard & mouse
- Frequently moving palm around keyboard keys
- Not using convenient editing features
  - Use ←, →, ↑, ↓ keys instead of Home, End, PageUp, PageDown
  - Use an editor with few convenient editing features
How can we edit text unproductively?

- Frequently moving hand between keyboard & mouse
- Frequently moving palm around keyboard keys
- Not using convenient editing features
  - Use ←, →, ↑, ↓ keys instead of Home, End, PageUp, PageDown
  - Use an editor with few convenient editing features

⇒ Emacs helps to avoid these problems!
What is Emacs?

- A text editor born in 1976 and still lives actively today
- The name Emacs stands for Editor MACroS
- The first Emacs version was written in 1976 by David A. Moon and Guy L. Steele Jr
- The most popular version GNU Emacs was created by Richard Stallman in 1984
- Many modern editors such as Eclipse, IntelliJ IDEA, Sublime, Visual Studio have an Emacs simulation mode
Emacs movements

• Use ←, →, ↑, ↓, Home, End, PageUp, PageDown as usual
• But b, f, p, n, a, e, v can avoid palm moving (b forward/f backward/p previous/n next/a...z/e end/v in)
• C-b, C-f: backward/forward character (C denotes the Ctrl key)
• C-p, C-n, C-a, C-e: previous/next line, begin/end of line
• C-v, M-v: scroll up/down window (M denotes the Alt key)
• M-b, M-f: backward/forward word
• Too complicated!
• But some are similar to Unix/Linux Shell!
Emacs movements

- Use ←, →, ↑, ↓, Home, End, PageUp, PageDown as usual
- But b, f, p, n, a, e, v can avoid palm moving (b: backward/ f: forward/ p: previous/ n: next/ a: ...z/ e: end/ v: in)
- C-b, C-f: backward/forward character (C denotes the Ctrl key)
- C-p, C-n, C-a, C-e: previous/next line, begin/end of line
- C-v, M-v: scroll up/down window (M denotes the Alt key)
- M-b, M-f: backward/forward word
- Too complicated!
- But some are similar to Unix/Linux Shell!
Emacs movements

- Use ←, →, ↑, ↓, Home, End, PageUp, PageDown as usual
- But b, f, p, n, a, e, v can avoid palm moving
  (backward/forward/previous/next /a ... z/end/v in ↓)
Emacs movements

- Use ←, →, ↑, ↓, Home, End, PageUp, PageDown as usual
- But b, f, p, n, a, e, v can avoid palm moving (backward/forward/previous/next/a...z/end/v in ↓)
  - C-b, C-f: backward/forward character (C denotes the Ctrl key)
Emacs movements

- Use ←, →, ↑, ↓, Home, End, PageUp, PageDown as usual
- But b, f, p, n, a, e, v can avoid palm moving (backward/forward/previous/next /a ... z/end/v in ↓)
  - C-b, C-f: backward/forward character (C denotes the Ctrl key)
  - C-p, C-n, C-a, C-e: previous/next line, begin/end of line
Emacs movements

- Use ←, →, ↑, ↓, Home, End, PageUp, PageDown as usual
- But b, f, p, n, a, e, v can avoid palm moving (backward/forward/previous/next /a ... z/end/v in ↓)
  - C-b, C-f: backward/forward character (C denotes the Ctrl key)
  - C-p, C-n, C-a, C-e: previous/next line, begin/end of line
  - C-v, M-v: scroll up/down window (M denotes the Alt key)
Emacs movements

- Use ←, →, ↑, ↓, Home, End, PageUp, PageDown as usual
- But b, f, p, n, a, e, v can avoid palm moving (backward/forward/previous/next /a ... z/end/v in ↓)
  - C-b, C-f: backward/forward character (C denotes the Ctrl key)
  - C-p, C-n, C-a, C-e: previous/next line, begin/end of line
  - C-v, M-v: scroll up/down window (M denotes the Alt key)
  - M-b, M-f: backward/forward word
Emacs movements

- Use ←, →, ↑, ↓, Home, End, PageUp, PageDown as usual
- But b, f, p, n, a, e, v can avoid palm moving (backward/forward/previous/next /a ... z/end/v in ↓)
  - C-b, C-f: backward/forward character (C denotes the Ctrl key)
  - C-p, C-n, C-a, C-e: previous/next line, begin/end of line
  - C-v, M-v: scroll up/down window (M denotes the Alt key)
  - M-b, M-f: backward/forward word

- Too complicated!
Emacs movements

• Use ←, →, ↑, ↓, Home, End, PageUp, PageDown as usual

• But b, f, p, n, a, e, v can avoid palm moving
  (backward/forward/previous/next /a ... z/end/v in ↓)
  ▶ C-b, C-f: backward/forward character (C denotes the Ctrl key)
  ▶ C-p, C-n, C-a, C-e: previous/next line, begin/end of line
  ▶ C-v, M-v: scroll up/down window (M denotes the Alt key)
  ▶ M-b, M-f: backward/forward word

• Too complicated!

• But some are similar to Unix/Linux Shell!
# Emacs movements

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-&lt;</td>
<td>Buffer</td>
</tr>
<tr>
<td>M-v</td>
<td>Window</td>
</tr>
<tr>
<td>M-d</td>
<td>Char</td>
</tr>
<tr>
<td>C-k</td>
<td>Word</td>
</tr>
<tr>
<td>C-y</td>
<td>End of Line</td>
</tr>
<tr>
<td>M-y</td>
<td>Paste</td>
</tr>
<tr>
<td>C-d</td>
<td>Copy</td>
</tr>
<tr>
<td>C-w</td>
<td>Cycle Kill Ring</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-p</td>
<td>Line</td>
</tr>
<tr>
<td>C-l</td>
<td>Char</td>
</tr>
<tr>
<td>M-\</td>
<td>Word</td>
</tr>
<tr>
<td>C-n</td>
<td>Line</td>
</tr>
<tr>
<td>C-s</td>
<td>Forward</td>
</tr>
<tr>
<td>M-C-s</td>
<td>Incremental Search</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-x r t</td>
<td>Insert</td>
</tr>
<tr>
<td>C-x r d</td>
<td>Delete</td>
</tr>
<tr>
<td>C-x r k</td>
<td>Kill</td>
</tr>
<tr>
<td>C-x r y</td>
<td>Yank Kill</td>
</tr>
<tr>
<td>C-x r r reg</td>
<td>Copy to reg</td>
</tr>
<tr>
<td>C-x r l reg</td>
<td>Insert from reg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-j</td>
<td>New line = indent</td>
</tr>
<tr>
<td>M-m</td>
<td>Move to Indentation</td>
</tr>
<tr>
<td>C-v</td>
<td>Window</td>
</tr>
<tr>
<td>M-\</td>
<td>Buffer</td>
</tr>
<tr>
<td>C-x 2</td>
<td>Split Horizontally</td>
</tr>
<tr>
<td>C-x 3</td>
<td>Split Vertically</td>
</tr>
<tr>
<td>C-x 0</td>
<td>Delete this window</td>
</tr>
<tr>
<td>C-x 1</td>
<td>Delete all other</td>
</tr>
<tr>
<td>C-x 0</td>
<td>Switch to other</td>
</tr>
</tbody>
</table>

---

By Martin Grand
Buffer and window

- **Emacs buffer**: The Emacs object containing text
- **Buffer** ≠ file: a file can be opened in multiple buffers
- **C-x C-f, C-x C-b, C-x k**: open file, switch buffer, kill buffer

- **Emacs window**: The Emacs object showing a buffer
- **Emacs' window** ≠ window in Linux/Windows (Emacs calls it frame)
- **C-x 0, 1, 2, 3**: delete, maximize, split horizontally/vertically
- **C-x {, }, ˆ, _**: shrink, enlarge horizontally/vertically
Buffer and window

• Emacs buffer:
  ▶ The Emacs object containing text
  ▶ Buffer ≠ file: a file can be opened in multiple buffers
  ▶ C-x C-f, C-x C-b, C-x k: open file, switch buffer, kill buffer
Buffer and window

- **Emacs buffer:**
  - The Emacs object containing text
  - Buffer \(\neq\) file: a file can be opened in multiple buffers
  - \(\text{C-x C-f, C-x C-b, C-x k}\): open file, switch buffer, kill buffer

- **Emacs window:**
  - The Emacs object showing a buffer
  - Emacs’ window \(\neq\) window in Linux/Windows (Emacs calls it frame)
  - \(\text{C-x 0, 1, 2, 3}\): delete, maximize, split horizontally/vertically
  - \(\text{C-x \{, \}, ^, _}\): shrink, enlarge horizontally/vertically
Mark, region and kill ring

- Mark and region
  - Mark is a special text position set by the command C-Space
  - Region is the text between the point (current cursor) and the mark
  - Flexibility: set mark once and adjust the region by moving the cursor

- Kill ring
  - The greedy clipboard of Emacs
  - M-w, C-w, C-y: copy, cut, yank (paste)
  - M-y: paste the previously copied/cut texts
  - C-k, M-d, C-Backspace, C-Del: ...store deleted texts into kill ring
Mark, region and kill ring

- Mark and region
  - Mark is a special text position set by the command C-Space
  - Region is the text between the point (current cursor) and the mark
  - Flexibility: set mark once and adjust the region by moving the cursor

- Kill ring
  - The greedy clipboard of Emacs
  - M-w, C-w, C-y: copy, cut, yank (paste)
  - M-y: paste the previously copied/cut texts
  - C-k, M-d, C-Backspace, C-Del: store deleted texts into kill ring
Mark, region and kill ring

- Mark and region
  - Mark is a special text position set by the command C-Space
  - Region is the text between the point (current cursor) and the mark
  - Flexibility: set mark once and adjust the region by moving the cursor

- Kill ring
  - The greedy clipboard of Emacs
  - M-w, C-w, C-y: copy, cut, yank (paste)
  - M-y: paste the previously copied/cut texts
  - C-k, M-d, C-Backspace, C-Del, ... store deleted texts into kill ring
Emacs, the advance
Advanced editing

• Edit, mark by symbolic expressions: smartparens
• Text completion: hippie-expand, dabbrev-expand
• Using macro: F3, F4
• Block editing: C-x SPC
• Text undo:
  ◀ Default undo: greedy, remember everything: undo of undo of undo ...
  ▶ Modern undo: undo-tree with visualization
Advanced editing

- Edit, mark by symbolic expressions: smartparens
Advanced editing

- Edit, mark by symbolic expressions: smartparens
- Text completion: hippie-expand, dabbrev-expand
Advanced editing

• Edit, mark by symbolic expressions: smartparens
• Text completion: hippie-expand, dabbrev-expand
• Using macro: F3, F4
Advanced editing

- Edit, mark by symbolic expressions: smartparens
- Text completion: hippie-expand, dabbrev-expand
- Using macro: F3, F4
- Block editing: C-x SPC
Advanced editing

- Edit, mark by symbolic expressions: smartparens
- Text completion: hippie-expand, dabbrev-expand
- Using macro: F3, F4
- Block editing: C-x SPC
- Text undo:
  - Default undo: greedy, remember everything: undo of undo of undo ...
  - Modern undo: undo-tree with visualization
Advanced editing

• Text conversion and transposition
  ▶ M-l, M-u, M-c: lowercase, uppercase, capitalize first character
  ▶ C-t, M-t, C-M-t, C-x C-t: transpose character, word, expression, line

• Text search, replace and highlighting in current buffer:
  ▶ Search: isearch, swivel, occur
  ▶ Replace: anzu
  ▶ Highlight: highlighting

Regular expression supported
Advanced editing

• Text conversion and transposition
  ▶ M-l, M-u, M-c: lowercase, uppercase, capitalize first character
  ▶ C-t, M-t, C-M-t, C-x C-t: transpose character, word, expression, line
Advanced editing

- Text conversion and transposition
  - M-1, M-u, M-c: lowercase, uppercase, capitalize first character
  - C-t, M-t, C-M-t, C-x C-t: transpose character, word, expression, line
- Text search, replace and highlighting in current buffer:
  - Search: isearch, swivel, occur
  - Replace: anzu
  - Highlight: highlighting
  - Regular expression supported
Handy commands
Handy commands

- Emacs commands completion and narrowing: helm, ivy

- Key-binding suggestion: which-key

- Text navigation in a few key strokes: avy

- Window selection: other-window, windmove, winum, winner

- Buffer navigation: helm-mini, previous/next buffer

- Working with other-window: <command>-other-window

- Others: just-one-space, delete-blank-lines, repeat
Handy commands

- Emacs commands completion and narrowing: helm, ivy
- Key-binding suggestion: which-key
Handy commands

• Emacs commands completion and narrowing: helm, ivy
• Key-binding suggestion: which-key
• Text navigation in a few key strokes: avy
Handy commands

- Emacs commands completion and narrowing: helm, ivy
- Key-binding suggestion: which-key
- Text navigation in a few key strokes: avy
- Window selection: other-window, windmove, winum, winner
Handy commands

- Emacs commands completion and narrowing: helm, ivy
- Key-binding suggestion: which-key
- Text navigation in a few key strokes: avy
- Window selection: other-window, windmove, winum, winner
- Buffer navigation: helm-mini, previous/next buffer
Handy commands

- Emacs commands completion and narrowing: helm, ivy
- Key-binding suggestion: which-key
- Text navigation in a few key strokes: avy
- Window selection: other-window, windmove, winum, winner
- Buffer navigation: helm-mini, previous/next buffer
- Working with other-window: <command>-other-window
Handy commands

- Emacs commands completion and narrowing: helm, ivy
- Key-binding suggestion: which-key
- Text navigation in a few key strokes: avy
- Window selection: other-window, windmove, winum, winner
- Buffer navigation: helm-mini, previous/next buffer
- Working with other-window: <command>-other-window
- Others: just-one-space, delete-blank-lines, repeat
Understand Emacs’ components

- Buffer, window, frame
- Major mode: display buffer, visualize and manipulate content
- Minor mode: auxiliary editing and visualizing features
- Mode line, header line: all auxiliary information
- Mini-buffer, echo area: user interacts with Emacs commands
Explore Emacs’ documentation

- Describe major mode: `C-h m (M-x describe-mode)`
- Describe key binding: `C-h k (M-x describe-key)`
- Describe function: `C-h f (M-x describe-function)`
- Describe variable: `C-h v (M-x describe-variable)`
Emacs, the ultimate
Integrated Development Environment

- Support almost every programming language:
  - E.g., C/C++, OCaml, Latex, Python, HTML, Java, LLVM bitcode
  - Comfortability: (almost) the same key bindings for all languages
- Syntax highlighting, auto indentation
- Code completion, code outline, jump to definition
- Lint checking (on-the-fly), syntax checking (on-the-fly)
- Compilation mode: jump back and forth to errors
- Weakness: code refactoring is limited
Integrated Development Environment

- Support almost every programming languages:
  - E.g., C/C++, OCaml, Latex, Python, HTML, Java, LLVM bitcode
  - Comfortability: (almost) the same key bindings for all languages
Integrated Development Environment

- Support almost every programming languages:
  - E.g., C/C++, OCaml, Latex, Python, HTML, Java, LLVM bitcode
  - Comfortability: (almost) the same key bindings for all languages
- Syntax highlighting, auto indentation

- Weakness: code refactoring is limited
Integrated Development Environment

• Support almost every programming languages:
  ▶ E.g., C/C++, OCaml, Latex, Python, HTML, Java, LLVM bitcode
  ▶ Comfortability: (almost) the same key bindings for all languages

• Syntax highlighting, auto indentation

• Code completion, code outline, jump to definition
Integrated Development Environment

- Support almost every programming languages:
  - E.g., C/C++, OCaml, Latex, Python, HTML, Java, LLVM bitcode
  - Comfortability: (almost) the same key bindings for all languages

- Syntax highlighting, auto indentation
- Code completion, code outline, jump to definition
- Lint checking (on-the-fly), syntax checking (on-the-fly)
Integrated Development Environment

• Support almost every programming languages:
  ➤ E.g., C/C++, OCaml, Latex, Python, HTML, Java, LLVM bitcode
  ➤ Comfortability: (almost) the same key bindings for all languages

• Syntax highlighting, auto indentation

• Code completion, code outline, jump to definition

• Lint checking (on-the-fly), syntax checking (on-the-fly)

• Compilation mode: jump back and forth to errors
Integrated Development Environment

- Support almost every programming languages:
  - E.g., C/C++, OCaml, Latex, Python, HTML, Java, LLVM bitcode
  - Comfortability: (almost) the same key bindings for all languages
- Syntax highlighting, auto indentation
- Code completion, code outline, jump to definition
- Lint checking (on-the-fly), syntax checking (on-the-fly)
- Compilation mode: jump back and forth to errors
- Weakness: code refactoring is limited
Integrated Development Environment

- **Project management:** speedbar, projectile
  - Navigate among files and folders
  - Search files in a project, search/replace text in files of a project
  - Switch between parent and child projects

- **Version control:** Git, Hg, CVS, Svn, Bazaar, Monotone, ...
  - **Git:** magit – everything that git cli can do
  - **Hg:** monky – influenced by magit

- **Terminal:** shell, term, eshell
Integrated Development Environment

- **Project management**: speedbar, projectile
  - Navigate among files and folders
  - Search files in a project, search/replace text in files of a project
  - Switch between parent and child projects

- **Version control**: Git, Hg, CVS, Svn, Bazaar, Monotone, ...
  - Git: magit – everything that git cli can do
  - Hg: monkey – influenced by magit
Integrated Development Environment

- Project management: speedbar, projectile
  - Navigate among files and folders
  - Search files in a project, search/replace text in files of a project
  - Switch between parent and child projects
- Version control: Git, Hg, CVS, Svn, Bazaar, Monotone, ...
  - Git: magit – everything that git cli can do
  - Hg: monky – influenced by magit
- Terminal: shell, term, eshell
Integrated Development Environment

- Project management: speedbar, projectile
  - Navigate among files and folders
  - Search files in a project, search/replace text in files of a project
  - Switch between parent and child projects

- Version control: Git, Hg, CVS, Svn, Bazaar, Monotone, ...
  - Git: magit – everything that git cli can do
  - Hg: monky – influenced by magit

- Terminal: shell, term, eshell

- Text diff and merge: diff, ediff
Everyday Emacs

- File manager:
  - Directory as a buffer: `dired`
  - Recent files, recent directories: `recentf`
- Text search, file search anywhere in the file system:
  - Text search: `grep`, `helm-grep`, `helm-ag`
  - File search: `helm-locate`
Everyday Emacs

- File manager:
  - Directory as a buffer: dired
  - Recent files, recent directories: recentf

Text search, file search anywhere in the file system

- Text search: grep, helm-grep, helm-ag
- File search: helm-locate
Everyday Emacs

- File manager:
  - Directory as a buffer: dired
  - Recent files, recent directories: recentf
- Text search, file search anywhere in the file system
  - Text search: grep, helm-grep, helm-ag
  - File search: helm-locate
Everyday Emacs

- Workspace management: eyebrowse
Everyday Emacs

- Workspace management: eyebrowse
- Note taking, project planning: org-mode
Everyday Emacs

- Workspace management: eyebrowse
- Note taking, project planning: org-mode
- Document viewing: images, pdf, presentation, archive files
Everyday Emacs

- Workspace management: eyebrowse
- Note taking, project planning: org-mode
- Document viewing: images, pdf, presentation, archive files
- Spell checking: aspell, hunspell, flyspell, langtool
Everyday Emacs

- Workspace management: eyebrowse
- Note taking, project planning: org-mode
- Document viewing: images, pdf, presentation, archive files
- Spell checking: aspell, hunspell, flyspell, langtool
- Web browser, remote access, email reader: eww, tramp, gnus
Everyday Emacs

- Workspace management: eyebrowse
- Note taking, project planning: org-mode
- Document viewing: images, pdf, presentation, archive files
- Spell checking: aspell, hunspell, flyspell, langtool
- Web browser, remote access, email reader: eww, tramp, gnus
- More handy, useful features, and even game, screensaver!
Getting started
## Emacs Starter Kits

<table>
<thead>
<tr>
<th>Name</th>
<th>Stars</th>
<th>Forks</th>
<th>Contributors</th>
<th>Commits</th>
<th>GitHub Sites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spacemacs</td>
<td>12.430</td>
<td>3.219</td>
<td>574</td>
<td>7.405</td>
<td>GH/syl20bnr/spacemacs</td>
</tr>
<tr>
<td>Purcell emacs</td>
<td>3.668</td>
<td>1.866</td>
<td>16</td>
<td>2.638</td>
<td>GH/purcell/emacs.d</td>
</tr>
<tr>
<td>Prelude</td>
<td>3.470</td>
<td>1.453</td>
<td>120</td>
<td>1.239</td>
<td>GH/bbatsov/prelude</td>
</tr>
<tr>
<td>Emacs live</td>
<td>1.240</td>
<td>245</td>
<td>34</td>
<td>703</td>
<td>GH/overtone/emacs-live</td>
</tr>
<tr>
<td>Magnars emacs</td>
<td>1.135</td>
<td>206</td>
<td>8</td>
<td>1.402</td>
<td>GH/magnars/.emacs.d</td>
</tr>
</tbody>
</table>

Updated in August 05\textsuperscript{th}, 2017
Create your own style Emacs

• Simple customization: GUI supported
  ▶ M-x customize-mode, M-x customize-group, etc

• Useful Emacs packages and libraries:
  ▶ https://github.com/emacs-tw/awesome-emacs

• Write your own Emacs functions or packages:
  ▶ Need to be familiar with Emacs Lisp language
  ▶ Starting point: copy an existing function and modify it
Create your own style Emacs

- Simple customization: GUI supported
  - M-x customize-mode, M-x customize-group, etc
Create your own style Emacs

- Simple customization: GUI supported
  - M-x customize-mode, M-x customize-group, etc
- Useful Emacs packages and libraries:
  - https://github.com/emacs-tw/awesome-emacs
Create your own style Emacs

• Simple customization: GUI supported
  ▶ M-x customize-mode, M-x customize-group, etc

• Useful Emacs packages and libraries:
  ▶ https://github.com/emacs-tw/awesome-emacs

• Write your own Emacs functions or packages:
  ▶ Need to be familiar with Emacs Lisp language
  ▶ Starting point: copy an existing function and modify it
Be aware

• Emacs pinky! Which key is used the most? How do you type it?
  ▶ Solution: use god-mode, hydra-mode, or a foot pedal!

• Emacs bankruptcy: `.emacs` will be out of control one day
  ▶ Solution: use Emacs starter kits

• Ctrl-Alt-Backspace: don’t try it on your Linux
  ▶ Solution: `sudo dpkg-reconfigure keyboard-configuration`

• Malicious/bad code from third-party packages
  ▶ Solution: do not use if they look suspicious

• Emacs customization may be addicted
  ▶ Solution: positive thinking – it could be fun!
Be aware

- Emacs pinky! Which key is used the most? How do you type it?

Solution: use god-mode, hydra-mode, or a foot pedal!

Solution: use Emacs starter kits

Solution: sudo dpkg-reconfigure keyboard-configuration

Solution: do not use if they look suspicious

Solution: positive thinking – it could be fun!
Be aware

- Emacs pinky! Which key is used the most? How do you type it?
  - Solution: use god-mode, hydra-mode, or a foot pedal!
Be aware

- Emacs pinky! Which key is used the most? How do you type it?
  - Solution: use god-mode, hydra-mode, or a foot pedal!
- Emacs bankruptcy: `.emacs` will be out of control one day
Be aware

- **Emacs pinky!** Which key is used the most? How do you type it?
  - Solution: use god-mode, hydra-mode, or a foot pedal!
- **Emacs bankruptcy:** `.emacs` will be out of control one day
  - Solution: use Emacs starter kits
Be aware

• Emacs pinky! Which key is used the most? How do you type it?
  ▶ Solution: use god-mode, hydra-mode, or a foot pedal!
• Emacs bankruptcy: .emacs will be out of control one day
  ▶ Solution: use Emacs starter kits
• Ctrl-Alt-Backspace: don’t try it on your Linux
Be aware

- Emacs pinky! Which key is used the most? How do you type it?
  - Solution: use god-mode, hydra-mode, or a foot pedal!
- Emacs bankruptcy: .emacs will be out of control one day
  - Solution: use Emacs starter kits
- Ctrl-Alt-Backspace: don’t try it on your Linux
  - Solution: sudo dpkg-reconfigure keyboard-configuration
Be aware

- Emacs pinky! Which key is used the most? How do you type it?
  ▶ Solution: use god-mode, hydra-mode, or a foot pedal!
- Emacs bankruptcy: .emacs will be out of control one day
  ▶ Solution: use Emacs starter kits
- Ctrl-Alt-Backspace: don’t try it on your Linux
  ▶ Solution: sudo dpkg-reconfigure keyboard-configuration
- Malicious/bad code from third-party packages
Be aware

- Emacs pinky! Which key is used the most? How do you type it?
  - Solution: use god-mode, hydra-mode, or a foot pedal!
- Emacs bankruptcy: .emacs will be out of control one day
  - Solution: use Emacs starter kits
- Ctrl-Alt-Backspace: don’t try it on your Linux
  - Solution: sudo dpkg-reconfigure keyboard-configuration
- Malicious/bad code from third-party packages
  - Solution: do not use if they looks suspicious
Be aware

• Emacs pinky! Which key is used the most? How do you type it?
  ▶ Solution: use god-mode, hydra-mode, or a foot pedal!
• Emacs bankruptcy: .emacs will be out of control one day
  ▶ Solution: use Emacs starter kits
• Ctrl-Alt-Backspace: don’t try it on your Linux
  ▶ Solution: sudo dpkg-reconfigure keyboard-configuration
• Malicious/bad code from third-party packages
  ▶ Solution: do not use if they looks suspicious
• Emacs customization may be addicted
Be aware

- Emacs pinky! Which key is used the most? How do you type it?
  - Solution: use god-mode, hydra-mode, or a foot pedal!
- Emacs bankruptcy: .emacs will be out of control one day
  - Solution: use Emacs starter kits
- Ctrl-Alt-Backspace: don’t try it on your Linux
  - Solution: sudo dpkg-reconfigure keyboard-configuration
- Malicious/bad code from third-party packages
  - Solution: do not use if they looks suspicious
- Emacs customization may be addicted
  - Solution: positive thinking – it could be fun!
Conclusion

• Introduce the versatility of Emacs and how to get started

• Advantages:
  ▶ Editor + file manager, project manager, terminal, multimedia, etc
  ▶ Big and active community

• Should you use Emacs?
  ▶ Depends on personal taste but Emacs is worth to be given a try!

Thank you!

Questions?
Conclusion

- Introduce the versatility of Emacs and how to get started
Conclusion

• Introduce the versatility of Emacs and how to get started
• Advantages:
Conclusion

• Introduce the versatility of Emacs and how to get started
• Advantages:
  ▶ Editor + file manager, project manager, terminal, multimedia, etc

Thank you!

Questions?
Conclusion

• Introduce the versatility of Emacs and how to get started

• Advantages:
  ▶ Editor + file manager, project manager, terminal, multimedia, etc
  ▶ Big and active community

Should you use Emacs?
▶ Depends on personal taste but Emacs is worth to be given a try!

Thank you!

Questions?
Conclusion

- Introduce the versatility of Emacs and how to get started
- Advantages:
  - Editor + file manager, project manager, terminal, multimedia, etc
  - Big and active community
- Should you use Emacs?

Thank you!

Questions?
Conclusion

• Introduce the versatility of Emacs and how to get started
• Advantages:
  ▶ Editor + file manager, project manager, terminal, multimedia, etc
  ▶ Big and active community
• Should you use Emacs?
  ▶ Depends on personal taste but Emacs is worth to be given a try!
Conclusion

• Introduce the versatility of Emacs and how to get started
• Advantages:
  ▶ Editor + file manager, project manager, terminal, multimedia, etc
  ▶ Big and active community
• Should you use Emacs?
  ▶ Depends on personal taste but Emacs is worth to be given a try!

Thank you!
Conclusion

• Introduce the versatility of Emacs and how to get started
• Advantages:
  ▶ Editor + file manager, project manager, terminal, multimedia, etc
  ▶ Big and active community
• Should you use Emacs?
  ▶ Depends on personal taste but Emacs is worth to be given a try!

Thank you!
Questions?