

Environnements et Outils de Développement

Cours 2 — Text editing with Emacs

Stefano Zacchioli

`zack@pps.univ-paris-diderot.fr`

Laboratoire PPS, Université Paris Diderot - Paris 7

URL <http://upsilon.cc/~zack/teaching/1112/ed6/>
Copyright © 2012 Stefano Zacchioli
License Creative Commons Attribution-ShareAlike 3.0 Unported License
<http://creativecommons.org/licenses/by-sa/3.0/>



It's all about text

Which tool, as programmer, will you use the most?

It's all about text

Which tool, as programmer, will you use the most?
A text editor.

- coding
- debugging
- searching / reading
- writing documentation
- configuration files
- (textual) program data
- ...

one Editor to rule them all

- complex software projects are often written using **multiple programming languages**
 - ▶ system lang. for the engine + application lang. for business logic
 - ▶ DSL
 - ▶ separation logic / presentation
 - ▶ documentation
- you'll work on several development projects at the “same time”

You'll change many languages ; **changing your editor** at each change wouldn't be wise.

Most programmers **choose one editor** and use it for every text editing task, no matter the language.

Which editor?

Doesn't matter, really, ... as long as it is flexible enough to adapt to your changing needs.

Two popular choices in the UNIX / Free Software world :

- Vim**
- **modal editor**; the editor responds differently to the same keys, depending on the editor state
 - **simple key strokes**, to be concatenated like video game “combos”
 - traditionally small and fast
 - used with **many instances** at a time

- Emacs**
- non-modal
 - **key combinations** and modifier keys (Ctrl+Alt+. . .)
 - traditionally **highly customizable**, using Emacs Lisp
 - use with a **single instance** running + clients

Biased !

See http://en.wikipedia.org/wiki/Editor_war

Choosing Emacs

For this class we got to choose (time constraints...).
And we've chosen Emacs.

Feel free to choose the one you like, really.

- learn to learn using your editor
- compare
- pick the one that makes you most efficient

Editing **concepts**, as well as coordination of other development utilities from your editor, are portable.

Tutorial

Programming major modes

- major modes to edit code in a specific programming language
- available for most (un)known programming languages
- auto-loaded based on file name extensions
- manually toggled by M-X *lang-mode*
 - ▶ c-mode
 - ▶ java-mode
 - ▶ caml-mode
 - ▶ python-mode
 - ▶ latex-mode
 - ▶ etc.

Common tasks

Most programming major modes support a common set of tasks, via a common interface.

- **indenting** : <TAB> is bound to (re-)indent the current line ; you'll use it *a lot* . .
- **commenting** : M-x comment-region, M-x uncomment-region
- **reformat paragraph** : M-q, mostly for text modes, but very useful in comments
- **completion** : M-/, trigger (word-based) completion, extensible
- **quickfix cycle** (edit/compile/fix)
Idea : **run an external compiler** and **parse its output** to detect errors and locate them in source code
 - ▶ M-x compile : compile (ask for compile command)
 - ▶ M-x recompile : compile (silent)
 - ▶ C-x ' : go to next error

Navigating through code

Generalize the idea behind the quickfix cycle

- **occur** : M-x occur navigate through **occurrences** in the current buffer
- **grep** : M-x grep navigate through occurrences, elsewhere

Rectangles

Consider point and mark, but look at the smallest **rectangular area of text** denoted by them

- C-x r k — kill rectangle
- C-x r y — yank rectangle
- C-x r o — open rectangle

Useful / alternative / non-standard way of re-indenting several lines at once.

Processing text with external tools

What if Emacs can't do a specific text manipulation that an external tool—more precisely a **UNIX filter**—could?

- 1 select the region you want to operate on
 - 2 `M-x shell-command-on-region`
bound by default to `M-|` (mnemonic for “pipe”, as on the shell)
- **read-only** ciao mode (default) : pipe the text to the filter, but do not change the text in return
 - **read-write** mode : pipe the text to the filter and replace it with filter's output
 - ▶ `C-u M-|`