Éditeurs de texte

Pierre Senellart

Semaine *Informatique pratique*, septembre 2026

Éditeurs de texte

- La programmation, la rédaction de documents LaTeX, la rédaction de preuve formelle pour assistants de preuve, le développement de sites Web… reposent sur l'édition de fichiers texte
- N'importe quel éditeur de texte fait l'affaire
- Grande variété :

  Éditeurs de base nano, pico, gedit, kate, Notepad, TextEdit, vi…

  Éditeurs intégrés à une application Web Overleaf

  Éditeurs d'un environnement de développement intégré TeXworks, TeXstudio, Eclipse, PyCharm, Visual Studio Code…

  Éditeurs génériques complexes vim, emacs, Sublime Text…
- Vous passerez des milliers d'heures dans votre éditeur : cela vaut la peine de choisir et d'apprendre

Éditeurs de texte

- La programmation, la rédaction de documents LaTeX, la rédaction de preuve formelle pour assistants de preuve, le développement de sites Web… reposent sur l'édition de fichiers texte
- N'importe quel éditeur de texte fait l'affaire
- Grande variété :

  Éditeurs de base nano, pico, gedit, kate, Notepad, TextEdit, vi…

  Éditeurs intégrés à une application Web Overleaf

  Éditeurs d'un environnement de développement intégré TeXworks, TeXstudio, Eclipse, PyCharm, Visual Studio Code…

  Éditeurs génériques complexes vim, emacs, Sublime Text…
- Vous passerez des milliers d'heures dans votre éditeur : cela vaut la peine de choisir et d'apprendre

Ce qui a changé : le Language Server Protocol

- Jusqu'aux années 2010, la prise en charge d'un langage (complétion, navigation, diagnostics) était réimplémentée dans chaque éditeur : les IDE spécialisés avaient un net avantage
- LSP (Language Server Protocol, 2016) : un serveur de langage (pyright pour Python, clangd pour C/C++, ocaml-lsp, le serveur de Lean 4, texlab pour LaTeX…) fournit ces services à n'importe quel éditeur disposant d'un client LSP
- Complétion sémantique, aller à la définition, renommage, références, reformatage, diagnostics : les mêmes dans tous les éditeurs, puisque c'est le même serveur
- Idem pour la coloration et la navigation structurelle (tree-sitter) et le débogage (DAP)
- Conséquence : vim, emacs, VS Code et les autres sont également capables pour Lean, Coq, OCaml, Python, LaTeX. Le choix d'un éditeur porte sur comment on aime éditer et travailler, pas sur ce que l'on édite

Tous proposent

Avant de regarder ce qui les distingue, ce que les éditeurs sérieux ont tous en commun :

- Coloration syntaxique
- Indentation automatique
- Fonctionnalités avancées de recherche, remplacement, manipulation de texte (en particulier basées sur des expressions rationnelles)
- Correction orthographique à la volée
- Un client LSP, donc tout ce qui précède : complétion sémantique, navigation, renommage, diagnostics
- Intégration avec outils externes (compilateurs, systèmes de contrôle de version, etc.)
- Intégration d'assistants fondés sur des modèles de langage (complétion, agents), nativement ou par extension
- Possibilité d'implémenter des extensions (typiquement en Vimscript pour vim, en Lua pour neovim, en Lisp pour emacs, en TypeScript pour VS Code)

vim

- vi : éditeur historiquement influent sous Unix, développé dans les années 70, qui s'est répandu sur toutes les variantes d'Unix
- vim : clone libre de vi datant de 1991, y apportant de nombreuses améliorations, toujours développé
- neovim : fork libre de vim datant des années 2010, configuré en Lua, client LSP et tree-sitter intégrés
- Basé sur la notion de mode d'édition : opérations différentes accessibles quand on écrit du texte (mode *insertion*), quand on sélectionne des morceaux de texte (mode *visuel*), quand on exécute des commandes (mode *commande*) ou le reste du temps (mode *normal*)
- Apparence relativement minimaliste, mais possibilités de configuration et d'extension très nombreuses
- Principalement utilisé au sein d'un terminal, même si des interfaces graphiques (gvim, neovide, etc.) existent

La grammaire de vim

Ce qui rend vim addictif : en mode normal, les commandes se composent comme une langue, opérateur + mouvement ou objet de texte.

`dw`	supprime (`d`) jusqu'au mot suivant (`w`)
`ci(`	change (`c`) l'intérieur (`i`) des parenthèses
`>ip`	indente (`>`) le paragraphe courant (`ip`)
`gUiw`	met en majuscules le mot sous le curseur
`3dd`	supprime trois lignes
`.`	répète la dernière modification
`qa...q`, `20@a`	enregistre une macro, la rejoue vingt fois
`:%s/x_(\d+)/y_\1/g`	substitution par expression rationnelle

- Apprendre une dizaine d'opérateurs et d'objets suffit pour être très efficace ; `vimtutor` en 30 minutes
- Cette grammaire est contagieuse : disponible dans VS Code (VSCodeVim), emacs (evil-mode), et reprise « à l'envers » (sélection d'abord, puis action) par Kakoune et Helix

emacs

- Éditeur phare (et libre) du projet GNU (le projet qui a fourni la plupart des utilitaires systèmes de Linux) datant des années 80, toujours développé
- Fait usage de nombreuses commandes claviers (avec les touches CTRL et Meta/Alt), dont certaines ont eu beaucoup d'influence dans d'autres contextes (shell, etc.)
- Intègre un interpréteur d'une version du langage fonctionnel Lisp, facilitant le développement de macros, d'extensions
- Approche assez maximaliste : l'éditeur de texte peut être l'environnement au sein duquel d'autres programmes tournent (par exemple, un logiciel de courrier électronique)
- En général lancé dans un environnement graphique, mais également utilisable à l'intérieur d'un terminal

emacs comme plateforme

Ce qui rend emacs addictif : tout est du Lisp, modifiable à chaud ; l'éditeur est une plateforme sur laquelle vivent des applications complètes.

magit	interface à git considérée par beaucoup comme la meilleure existante
org-mode	notes, listes de tâches, agenda, documents « littéraires » mêlant texte et code exécutable
TRAMP	édition transparente de fichiers distants (`/ssh:machine:/chemin`), y compris via des rebonds
dired, shell, eshell	gestionnaire de fichiers et terminaux intégrés
eglot, lsp-mode	clients LSP (eglot est intégré depuis emacs 29)
AUCTEX	l'environnement LaTeX de référence

Macros clavier : `F3`. . .`F4` pour enregistrer, `F4` pour rejouer.
Configurations prêtes à l'emploi : Doom Emacs, Spacemacs.

Visual Studio Code

- Logiciel propriétaire mais gratuit, développé par Microsoft, disponible pour Windows, macOS et Linux, basé en très grande partie sur un logiciel libre (VSCodium en est une distribution entièrement libre, sans télémétrie)
- Date des années 2010, évolution des environnements propriétaires spécifiques à certains langages que Microsoft proposait auparavant
- Logiciel moderne, configurable, extensible, devenu très populaire parmi les développeurs ; c'est de lui que vient LSP
- Revers de la médaille : nettement plus lourd que vim ou emacs
- Uniquement disponible sous forme d'une interface graphique, fait pour être contrôlé à la souris (mais nombreux raccourcis claviers)

VS Code en pratique

Palette de commandes `Ctrl+Shift+P` : toute commande accessible par son nom, sans mémoriser de raccourci

Multi-curseurs `Ctrl+D` sélectionne l'occurrence suivante du mot, `Ctrl+Shift+L` toutes les occurrences, `Alt+clic` ajoute un curseur : on édite plusieurs endroits à la fois là où vim ou emacs utiliseraient une macro

Extensions un catalogue immense (LaTeX Workshop, Python, Lean 4, VsCoq, OCaml Platform…), installées en un clic

Remote – SSH, conteneurs l'interface tourne localement, un serveur VS Code s'exécute sur la machine distante (ou dans un conteneur) ; confortable, mais suppose de pouvoir y lancer ce serveur

Live Share édition collaborative en temps réel

Terminal intégré et intégration git de base

La même tâche dans les trois

	neovim	emacs	VS Code
Aller à la définition	`Ctrl+]`	`M-.`	`F12`
Renommer un symbole	`grn`	`M-x eglot-rename`	`F2`
Références	`grr`	`M-?`	`Shift+F12`
Commenter des lignes	`gc` + mouvement	`M-;`	`Ctrl+/`
Remplacement par expression rationnelle	`:%s/a/b/g`	`C-M-%`	`Ctrl+H`, bouton `.*`
Modifier dix endroits semblables	macro `qa...q`, `10@a`	macro `F3...F4`, `C-u 10 F4`	multi-curseurs `Ctrl+D`
Fichier distant	via SSH + tmux	TRAMP	Remote – SSH

Les trois premières lignes viennent du serveur LSP : même fonctionnalité, même qualité, seul le raccourci change. (Raccourcis par défaut de neovim $\geq$ 0.11 et d'eglot.)

Outils indépendants de l'éditeur

Comme les serveurs LSP, beaucoup d'outils se configurent une fois et servent dans tous les éditeurs :

Formateurs	black ou ruff (Python), clang-format (C/C++), ocamlformat, latexindent : un style uniforme sans y penser
Linters	ruff, clang-tidy, chktex (LaTeX) : erreurs et mauvaises pratiques signalées au fil de l'écriture
EditorConfig	fichier `.editorconfig` à la racine d'un projet fixant indentation, encodage, fins de ligne pour tous les contributeurs, quel que soit leur éditeur
Correcteurs	hunspell, aspell (et leurs dictionnaires)
Débogueurs	gdb, lldb, debugpy, via DAP

Investir dans ces outils est rentable même si vous changez d'éditeur.

Comment choisir ?

Quelques questions à se poser :

- Terminal ou graphique ? Un éditeur en mode texte fonctionne identiquement en local, en SSH, dans tmux, sur un serveur de calcul
- Clavier ou souris ? L'édition modale demande un apprentissage, puis on ne quitte plus le clavier
- Configurer ou pas ? VS Code marche bien « out of the box » ; vim et emacs récompensent ceux qui investissent dans leur configuration (ou partent d'une configuration existante : LazyVim, kickstart.nvim, Doom Emacs)
- Où vivent vos autres outils ? git, courrier, notes, terminaux : dans l'éditeur (emacs), à côté (vim + tmux + shell) ou dans une application graphique
- Avec qui travaillez-vous ? Édition simultanée (Live Share, Overleaf) ou asynchrone via git

Le langage utilisé n'est pas un critère : grâce à LSP, tous se valent.

Prenez le temps de choisir !

- Un bon éditeur de texte vous fera gagner du temps, éviter des erreurs (d'orthographe, de syntaxe, etc.)
- Important de trouver un confort dans son éditeur, au sein duquel vous passerez une bonne partie de vos études et de votre carrière
- Dommage de se contenter d'un mauvais éditeur, juste par flemme d'en maîtriser un meilleur
- Un logiciel graphique comme VS Code peut être plus facile d'accès que vim ou emacs, mais ça ne veut pas dire qu'il est le meilleur pour vous
- Deux pièges symétriques : l'inertie (rester sous nano ou dans l'éditeur Web d'Overleaf) et le puits sans fond de la configuration
- Pour essayer : `vimtutor`, le tutoriel d'emacs (`C-h t`), la page d'accueil interactive de VS Code ; puis le même fichier pendant vingt minutes dans chacun