GUIDE PÉDAGOGIQUE — NIVEAU DÉBUTANT

VIM

Apprendre VIM de zéro

Le cours complet pour comprendre, prendre en main et commencer à utiliser l'éditeur de texte VIM — expliqué pas à pas, sans prérequis.

13
chapitres progressifs

100+
commandes expliquées

0
prérequis nécessaire

Table des matières

Annexes

Comment lire ce cours ?

Ce guide est **progressif** : chaque chapitre s'appuie sur le précédent. Lis-le dans l'ordre, et surtout **ouvre VIM en parallèle** pour tester chaque commande. VIM s'apprend avec les doigts, pas seulement avec les yeux. Les touches du clavier sont notées ainsi : `i`, `Echap`, `:wq`.

CHAPITRE 1

Introduction : qu'est-ce que VIM ?

Avant de taper la moindre touche, comprenons ce qu'est VIM, pourquoi tant de développeurs l'utilisent, et en quoi il est radicalement différent des éditeurs habituels.

Un éditeur de texte, mais pas comme les autres

VIM (contraction de *Vi IMproved*, « Vi amélioré ») est un éditeur de texte qui fonctionne directement dans le terminal. Il est l'héritier modernisé de *vi*, un éditeur né en 1976. Autrement dit, VIM est un outil mûri par près de cinquante ans d'usage intensif.

Contrairement à un traitement de texte comme Word ou à un éditeur moderne comme VS Code, VIM ne se pilote pas principalement à la souris. **Tout se fait au clavier**, grâce à un langage de commandes très court et très précis. C'est déroutant au début, et extraordinairement efficace une fois acquis.

> **Vi, VIM, Neovim : qui est qui ?**
> **vi** est l'ancêtre, présent sur presque tous les systèmes Unix. **VIM** est sa version moderne et largement répandue, et c'est celle de ce cours. **Neovim** est une réécriture récente de VIM, très populaire, qui partage l'essentiel des commandes : tout ce que vous apprenez ici fonctionnera aussi dans Neovim.

Pourquoi apprendre VIM en 2026 ?

- **Il est partout.** Sur quasiment tout serveur Linux ou Mac, `vi` ou `vim` est déjà installé. Savoir l'utiliser, c'est pouvoir éditer un fichier *n'importe où*, même sur une machine distante via SSH où aucun autre éditeur n'est disponible.
- **Les mains restent sur le clavier.** Pas d'aller-retour vers la souris : on gagne en vitesse et en confort sur le long terme.
- **Il est léger et rapide.** VIM démarre instantanément et gère sans broncher des fichiers énormes.
- **Ses commandes se retrouvent ailleurs.** De nombreux outils (le shell, les pagers comme `less`, des extensions pour VS Code, etc.) reprennent les raccourcis de VIM.

La grande idée : l'édition « modale »

Voici le concept le plus important de tout ce cours. Dans un éditeur classique, votre clavier sert **toujours** à écrire : appuyer sur `d` écrit la lettre « d ».

VIM, lui, est **modal** : selon le *mode* dans lequel vous êtes, une même touche fait des choses différentes. En mode **Insertion**, `d` écrit « d ». Mais en mode **Normal**, `d` devient un ordre : « supprime ». Le clavier tout entier se transforme en télécommande d'édition.

À retenir

VIM ne sépare pas « écrire » et « commander » par des touches comme `Ctrl` ou `Cmd`, mais par des **modes**. C'est exactement ce que nous étudions au chapitre 3 — et c'est la clé qui débloque tout le reste.

Soyons honnêtes : la courbe d'apprentissage

VIM a la réputation d'être difficile. La vérité est plus nuancée : les **premières heures** sont déroutantes (« comment est-ce que je quitte ?! »), puis tout s'éclaire très vite. Ce cours est justement conçu pour franchir cette première marche sans douleur. Soyez patient avec vous-même : c'est normal d'être lent au début.

CHAPITRE 2

Installer et lancer VIM

Mettons VIM en marche sur votre machine, puis découvrons l'écran de démarrage et le tutoriel intégré qui vous accompagnera.

Installer VIM

VIM est souvent déjà présent. Pour vérifier, ouvrez un terminal et tapez :

```
$ vim --version
```

Si une version s'affiche, VIM est installé. Sinon, voici comment l'obtenir :

Système	Commande d'installation
Linux (Debian / Ubuntu)	`sudo apt install vim`
Linux (Fedora)	`sudo dnf install vim`
Linux (Arch)	`sudo pacman -S vim`
macOS	Déjà installé. Pour la dernière version : `brew install vim`
Windows	Via `winget install vim.vim`, ou télécharger l'installeur sur le site officiel `vim.org`

> **Astuce Windows**
> Sur Windows, la façon la plus simple et la plus utile d'apprendre VIM est de travailler dans **WSL** (le sous-système Linux de Windows). Vous y retrouverez l'environnement exact d'un serveur réel.

Lancer VIM

Depuis le terminal, deux usages courants :

```
$ vim              # ouvre VIM sur un écran vide
$ vim mon_fichier.txt   # ouvre (ou crée) ce fichier
```

L'écran de démarrage

En lançant `vim` seul, vous voyez un écran d'accueil avec le nom du programme et quelques indications. Les colonnes de tildes `~` sur la gauche indiquent simplement les lignes **vides** situées après la fin du fichier — ce n'est pas du texte, juste un repère visuel.

Tout en bas se trouve la **ligne d'état** : c'est là que VIM affiche le nom du fichier, votre position, et les commandes que vous tapez. Prenez l'habitude de la regarder, elle vous renseigne en permanence.

vimtutor : votre meilleur ami

VIM est livré avec un tutoriel interactif de 25 à 30 minutes, dans votre langue. C'est la meilleure façon de mettre en pratique ce cours. Quittez VIM, et dans le terminal tapez :

```
$ vimtutor
```

Un vrai fichier d'exercices s'ouvre : vous lisez une consigne, puis vous l'appliquez directement dans le texte. Faites-le au moins une fois en complément de ce guide.

Le plan d'attaque conseillé

1) Lisez les chapitres 3 et 4 de ce cours (modes + survie). 2) Faites `vimtutor` en entier. 3) Revenez ici pour les chapitres 5 à 11. 4) Utilisez VIM pour de vraies petites tâches. La pratique régulière vaut mieux que de longues sessions.

CHAPITRE 3

Le concept clé : les modes

C'est LE chapitre fondateur. Si vous comprenez bien les modes, 90 % de la confusion autour de VIM disparaît.

Les quatre modes à connaître

VIM possède plusieurs modes ; pour débuter, quatre suffisent largement :

NORMAL
Le mode par défaut. On s'y déplace et on y donne des ordres d'édition.
On n'écrit PAS ici.

INSERTION
Le seul mode où l'on tape réellement du texte, comme dans un éditeur classique.

VISUEL
Pour sélectionner du texte (comme avec la souris) puis agir dessus.

COMMANDE
Pour les ordres tapés sur la ligne du bas, comme enregistrer ou quitter.

Le mode Normal : le point de départ

Quand VIM démarre, vous êtes en **mode Normal**. C'est contre-intuitif au début : vous ne pouvez pas écrire directement ! Si vous tapez « bonjour », VIM va interpréter chaque lettre comme une commande et faire des choses étranges. C'est normal.

Le mode Normal est le **carrefour central** de VIM. On y revient sans cesse. Depuis lui, on passe vers les autres modes, et on y retourne toujours après une action.

Passer en mode Insertion pour écrire

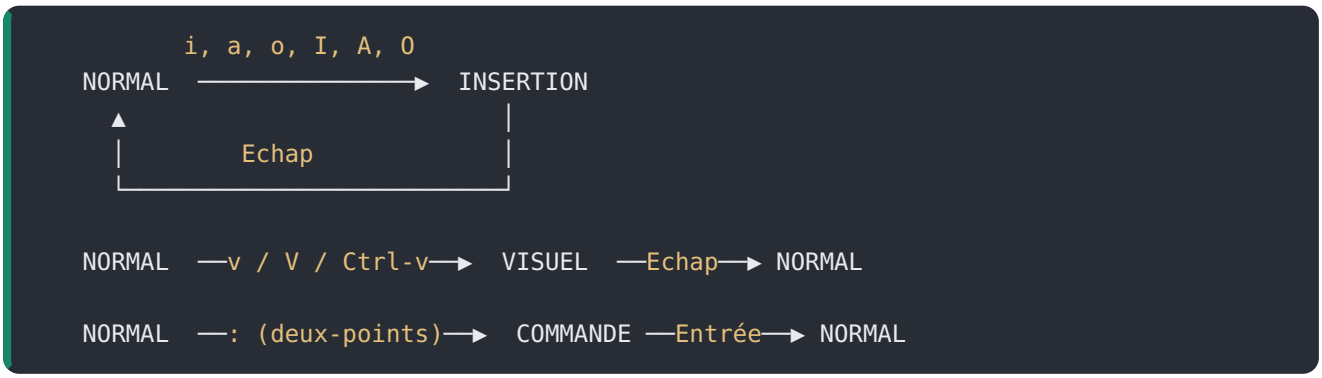
Pour taper du texte, on entre en mode Insertion. La touche la plus simple est `i` (pour *insert*). Une fois dedans, VIM se comporte comme un éditeur normal et la mention `-- INSERTION --` apparaît en bas de l'écran.

La touche la plus importante de VIM : Échap

À retenir absolument

Pour **sortir** du mode Insertion (ou de presque n'importe quel mode) et **revenir au mode Normal**, on appuie sur `Echap` (Escape). En cas de doute, si vous êtes perdu ou que VIM réagit bizarrement : **appuyez sur** `Echap`. Vous reviendrez au point neutre.

Schéma : la circulation entre les modes



Retenez la logique : on **part toujours du mode Normal**, on **entre** dans un autre mode par une touche dédiée, et on **revient** au Normal avec `Echap` (ou `Entrée` pour le mode Commande).

> **Erreur classique du débutant**
>
> Taper du texte sans être passé en mode Insertion. Si des caractères « disparaissent » ou déclenchent des actions étranges, c'est que vous étiez resté en mode Normal. Réflexe : `Echap`, puis `u` pour annuler les dégâts (voir chapitre 9), puis `i` pour écrire.

Comment savoir dans quel mode je suis ?

Regardez la **ligne d'état**, tout en bas à gauche : en mode Insertion, VIM affiche `-- INSERTION --` ; en mode Visuel, `-- VISUEL --`. Si rien n'est affiché, vous êtes en mode Normal.

CHAPITRE 4

Le kit de survie : ouvrir, enregistrer, quitter

« Comment je sors de VIM ?! » est la question la plus célèbre d'Internet. Réglons-la définitivement, ainsi que tout ce qu'il faut pour ne jamais rester coincé.

Le mode Commande (les deux-points)

Enregistrer et quitter passent par le **mode Commande**. Depuis le mode Normal, appuyez sur `:` (deux-points). Le curseur saute en bas de l'écran et vous pouvez taper une commande, puis valider avec `Entrée`.

Le piège n°1

Si `:` ne fait rien ou écrit un caractère dans le texte, c'est que vous êtes en mode Insertion. Appuyez d'abord sur `Echap` pour revenir en mode Normal, **puis** sur `:`.

Les commandes essentielles de sortie

Commande	Effet
`:w`	**Enregistrer** (write) le fichier sans quitter
`:q`	**Quitter** (quit) — refusé s'il y a des modifications non enregistrées
`:wq`	**Enregistrer puis quitter** (le plus courant)
`:x`	Comme `:wq`, mais n'enregistre que si nécessaire
`:q!`	Quitter **en forçant**, sans enregistrer (on abandonne les modifications)
`:wq!`	Forcer l'enregistrement puis quitter (utile pour un fichier en lecture seule)
`ZZ`	Raccourci en mode Normal (sans `:`) = enregistrer et quitter
`ZQ`	Raccourci en mode Normal = quitter sans enregistrer

La recette anti-blocage

Vous êtes perdu et voulez juste sortir ? Tapez dans l'ordre, lentement : `Echap` puis `:` `q` `!` puis `Entrée`. Cela quitte sans rien enregistrer, quoi qu'il arrive.

Le point d'exclamation : « force »

Vous l'avez remarqué : le `!` signifie « **force-le** ». VIM vous protège en refusant de quitter si vous avez des changements non sauvegardés (il affiche un avertissement comme `E37`). Le `!` lui dit : « je sais ce que je fais, exécute quand même ».

Ouvrir et gérer les fichiers

Commande	Effet
`:e fichier`	Ouvrir (edit) un autre fichier
`:w nom`	Enregistrer **sous** un autre nom
`:saveas nom`	Enregistrer sous et continuer à éditer le nouveau fichier

Pratiquons tout de suite

Voici votre premier exercice complet, du début à la fin :

```
$ vim essai.txt        # 1. ouvrir un nouveau fichier
# 2. appuyer sur  i        -> passe en INSERTION
# 3. taper :  Bonjour VIM !
# 4. appuyer sur  Echap    -> retour en NORMAL
# 5. taper :  :wq  puis Entrée  -> enregistre et quitte
$ cat essai.txt        # vérifie : "Bonjour VIM !" s'affiche
```

Si vous réussissez cet enchaînement, vous savez déjà faire l'essentiel : entrer, écrire, sauvegarder, sortir. Tout le reste n'est que de la puissance en plus.

CHAPITRE 5

Se déplacer dans le texte

En mode Normal, votre clavier devient une télécommande de navigation. Apprenez à bouger sans toucher les flèches ni la souris : c'est là que VIM commence à payer.

Les quatre touches de base : h j k l

Plutôt que les flèches, VIM utilise quatre lettres voisines pour déplacer le curseur. Vos doigts restent sur la rangée centrale du clavier.

Touche	Direction	Moyen mnémotechnique
`h`	← Gauche	La touche la plus à gauche du groupe
`j`	↓ Bas	Le « j » a une jambe qui descend
`k`	↑ Haut	Juste au-dessus de « j »
`l`	→ Droite	La touche la plus à droite du groupe

Astuce d'apprentissage

Les flèches du clavier fonctionnent aussi, mais forcez-vous à utiliser `h` `j` `k` `l` dès le départ. C'est inconfortable une journée, puis cela devient un réflexe qui vous fera gagner un temps fou.

Se déplacer par mots

Bouger lettre par lettre est lent. VIM permet de sauter de mot en mot :

Touche	Effet
`w`	Avancer au **début du mot suivant** (word)
`b`	Reculer au **début du mot précédent** (back)
`e`	Avancer à la **fin du mot** (end)

Se déplacer sur une ligne

Touche	Effet
`0`	Aller au **tout début** de la ligne (colonne 0)
`^`	Aller au **premier caractère non vide** de la ligne
`$`	Aller à la **fin** de la ligne

Se déplacer dans le fichier entier

Touche	Effet
`gg`	Aller tout en **haut** du fichier
`G`	Aller tout en **bas** du fichier
`42G` ou `:42`	Aller directement à la **ligne 42**
`Ctrl`+`d`	Descendre d'un **demi-écran** (down)
`Ctrl`+`u`	Monter d'un **demi-écran** (up)
`Ctrl`+`f`	Avancer d'un **écran complet** (forward)
`Ctrl`+`b`	Reculer d'un **écran complet** (back)

Sauter sur un caractère précis dans la ligne

Très pratique pour viser un endroit précis : `f` suivi d'un caractère place le curseur sur la **prochaine occurrence** de ce caractère dans la ligne. Par exemple, `f(` saute à la prochaine parenthèse ouvrante.

Commande	Effet
`f`x	Aller **sur** la prochaine occurrence du caractère x (find)
`t`x	Aller **juste avant** la prochaine occurrence de x (till)
`;`	Répéter le dernier `f` ou `t`

Le préfixe numérique

Faites précéder presque n'importe quel mouvement d'un **nombre** pour le répéter. `5j` descend de 5 lignes, `3w` avance de 3 mots, `10l` avance de 10 caractères. Ce principe « **nombre + commande** » est central dans VIM et revient partout.

CHAPITRE 6

Saisir du texte (mode Insertion)

Il existe plusieurs portes d'entrée vers le mode Insertion, chacune plaçant le curseur au bon endroit. Les connaître évite de se déplacer inutilement avant d'écrire.

Les commandes d'entrée en Insertion

Toutes ces touches s'utilisent **depuis le mode Normal** et vous basculent en Insertion, mais à des positions différentes :

Touche	Où l'on commence à écrire
`i`	**Avant** le curseur (insert)
`a`	**Après** le curseur (append) — idéal pour ajouter à la suite d'une lettre
`I`	Au **début** de la ligne (premier caractère non vide)
`A`	À la **fin** de la ligne — très utilisé
`o`	Ouvre une **nouvelle ligne en dessous** et y place le curseur
`O`	Ouvre une **nouvelle ligne au-dessus**

La logique majuscule / minuscule

Remarquez le motif : la version **minuscule** agit « localement / avant », la version **MAJUSCULE** agit « en grand / à la ligne / après ». `a` ajoute après le curseur, `A` ajoute en fin de ligne. Ce parallèle minuscule/MAJUSCULE se retrouve dans beaucoup de commandes VIM.

Un exemple concret

Imaginons cette ligne, le curseur (▮) étant sur le « m » de « monde » :

```
Bonjour le monde
```

- `i` puis « beau » donne : `Bonjour le beaumonde` (insère avant le m)
- `A` puis « !» donne : `Bonjour le monde !` (saute en fin de ligne)
- `o` puis « Deuxième ligne » crée une nouvelle ligne en dessous prête à recevoir le texte

Ne pas oublier

Après avoir écrit, **revenez toujours en mode Normal avec** `Echap`. Tant que vous restez en Insertion, vous ne pouvez ni vous déplacer efficacement, ni enregistrer, ni donner de commandes.

CHAPITRE 7

Supprimer, copier, coller

Voici les opérations d'édition de tous les jours. Elles se font en mode Normal et préparent la « grammaire » que nous verrons au chapitre suivant.

Supprimer

Commande	Effet
`x`	Supprimer **le caractère** sous le curseur
`dw`	Supprimer **du curseur jusqu'à la fin du mot** (delete word)
`dd`	Supprimer **la ligne entière**
`d$`	Supprimer **du curseur jusqu'à la fin de la ligne**
`3dd`	Supprimer **3 lignes** d'un coup

« Supprimer » = « couper »

Dans VIM, supprimer ne jette pas le texte à la poubelle : il est placé dans une mémoire temporaire (un « registre »). C'est en réalité un **couper**. Vous pourrez donc le recoller ailleurs avec `p`.

Copier (« yank »)

Dans le vocabulaire de VIM, copier se dit **yank**, d'où la lettre `y`.

Commande	Effet
`yy`	Copier **la ligne entière**
`yw`	Copier **jusqu'à la fin du mot**
`y$`	Copier **jusqu'à la fin de la ligne**
`2yy`	Copier **2 lignes**

Coller (« paste »)

Commande	Effet
`p`	Coller **après** le curseur (ou sous la ligne, si on a copié une ligne)
`P`	Coller **avant** le curseur (ou au-dessus de la ligne)

Remplacer et modifier rapidement

Commande	Effet
`r`x	**Remplacer** le caractère sous le curseur par x, sans changer de mode
`cw`	**Changer** le mot : le supprime ET passe en Insertion (change word)
`cc`	Changer toute la ligne (la vide et passe en Insertion)
`C`	Changer du curseur jusqu'à la fin de la ligne

Le combo le plus utilisé au monde

`dd` pour couper une ligne, se déplacer où l'on veut, puis `p` pour la recoller. C'est ainsi qu'on **déplace une ligne** en deux temps. Essayez-le, vous l'utiliserez tous les jours.

CHAPITRE 8

La grammaire de VIM : opérateur + mouvement

Voici le secret qui transforme une poignée de commandes en un langage quasi infini. Comprenez ce chapitre et vous « parlerez » VIM, au lieu de mémoriser des raccourcis par cœur.

VIM est un langage, pas une liste de raccourcis

Vous avez peut-être remarqué une régularité : `dw` = supprimer un mot, `yw` = copier un mot, `cw` = changer un mot. Ce n'est pas un hasard. Les commandes de VIM se construisent comme une **phrase** :

```
[nombre]  +  opérateur  +  mouvement / objet

exemple :   2dw   =   « supprime 2 mots »
            d$    =   « supprime jusqu'à la fin de ligne »
            ygg   =   « copie jusqu'au début du fichier »
```

Les « verbes » : les opérateurs

Un opérateur dit *quoi faire*. Les trois principaux :

Opérateur	Action (le « verbe »)
`d`	**delete** — supprimer (couper)
`y`	**yank** — copier
`c`	**change** — remplacer (supprime puis passe en Insertion)

Les « compléments » : les mouvements

Le mouvement dit *sur quelle portion de texte* appliquer le verbe. Bonne nouvelle : ce sont les déplacements du chapitre 5 que vous connaissez déjà !

Mouvement	Portion concernée
`w`	jusqu'au mot suivant
`$`	jusqu'à la fin de la ligne
`0`	jusqu'au début de la ligne
`G`	jusqu'à la fin du fichier
`f`x	jusqu'au caractère x

La règle qui fait tout

N'importe quel **opérateur** se combine avec n'importe quel **mouvement**. Si vous connaissez 3 opérateurs et 8 mouvements, vous connaissez déjà 24 commandes. Vous n'apprenez pas des raccourcis, vous apprenez un **système combinatoire**.

Les objets texte : viser intelligemment

VIM va plus loin avec les **objets texte**, qui désignent une structure entière, peu importe où se trouve le curseur à l'intérieur. Ils se forment avec `i` (*inner*, l'intérieur) ou `a` (*around*, avec les délimiteurs).

Objet	Désigne	Exemple
`iw`	le mot (inner word)	`diw` = supprimer le mot entier
`i"`	l'intérieur des guillemets	`ci"` = changer le texte entre " "
`i(`	l'intérieur des parenthèses	`di(` = vider la parenthèse
`ip`	le paragraphe	`dip` = supprimer le paragraphe
`a"`	les guillemets **inclus**	`da"` = supprimer " et leur contenu

Exemple bluffant

Curseur n'importe où entre deux guillemets : `nom = "ancien texte"`. Tapez `c` `i` `"` : tout le contenu entre les guillemets disparaît et vous pouvez taper le nouveau texte. Pas besoin de viser précisément le début ou la fin.

Le doublement : agir sur la ligne entière

Quand on **double l'opérateur**, il s'applique à la ligne courante : `dd` supprime la ligne, `yy` la copie, `cc` la change. C'est pour cela que ces commandes vous semblaient « spéciales » au chapitre 7 — en réalité elles suivent la même grammaire.

CHAPITRE 9

Annuler, refaire, répéter

Trois commandes qui vous donnent un filet de sécurité et un superpouvoir de productivité. Sans elles, on n'ose pas expérimenter ; avec elles, on travaille sereinement.

Annuler et refaire

Commande	Effet
`u`	**Annuler** la dernière modification (undo). On peut appuyer plusieurs fois.
`Ctrl`+`r`	**Rétablir** ce qu'on vient d'annuler (redo)
`U`	Annuler **toutes** les modifications faites sur la ligne courante

N'ayez plus peur d'expérimenter

Comme `u` annule presque tout, vous pouvez tester librement des commandes que vous ne maîtrisez pas encore. Une bêtise ? `u`, et c'est comme si rien ne s'était passé. C'est la meilleure façon d'apprendre vite.

La commande la plus magique : le point

La touche `.` (le point) **répète la dernière modification**. C'est l'une des fonctionnalités les plus aimées de VIM.

Exemple : vous supprimez un mot avec `dw`. Déplacez-vous sur un autre mot, appuyez sur `.` : il est supprimé aussi, sans retaper la commande. Sur un troisième mot, `.` encore. Vous venez de répéter une action complexe en une seule frappe.

Le duo gagnant

`.` (répéter) combiné aux mouvements (chapitre 5) et à la recherche (chapitre 10) forme la base du flux de travail rapide en VIM : *se déplacer → faire une modification → se déplacer → répéter avec* `.`. Gardez ce schéma en tête.

CHAPITRE 10

Rechercher et remplacer

Retrouver un mot dans un fichier de 5000 lignes, ou remplacer toutes les occurrences d'un terme d'un coup : ces commandes vous font gagner un temps considérable.

Rechercher

Depuis le mode Normal, tapez `/` suivi du texte cherché, puis `Entrée`. VIM saute à la première occurrence.

Commande	Effet
`/`mot	Chercher « mot » **vers l'avant**
`?`mot	Chercher « mot » **vers l'arrière**
`n`	Occurrence **suivante** (next)
`N`	Occurrence **précédente**
`*`	Chercher le mot **sous le curseur** directement

Remplacer (la commande :s)

Le remplacement utilise le mode Commande avec `:s` (substitute). Sa structure générale :

```
:s/ancien/nouveau/options
```

Commande	Effet
`:s/foo/bar/`	Remplacer la **1re** occurrence de « foo » par « bar » **sur la ligne**
`:s/foo/bar/g`	Remplacer **toutes** les occurrences **de la ligne** (g = global)
`:%s/foo/bar/g`	Remplacer toutes les occurrences **de tout le fichier** (% = tout le fichier)
`:%s/foo/bar/gc`	Pareil, mais en **demandant confirmation** à chaque fois (c = confirm)

Décortiquons :%s/foo/bar/g

`%` = « sur tout le fichier » · `s` = « substitue » · `foo` = ce qu'on cherche · `bar` = ce qu'on met à la place · `g` = « toutes les occurrences, pas juste la première ». Une fois ces briques comprises, vous composerez vos propres remplacements.

Conseil de prudence

Sur un remplacement global important, ajoutez l'option `c` (`:%s/foo/bar/gc`) pour valider chaque remplacement. VIM vous demandera `y` (oui), `n` (non), `a` (tout), `q` (arrêter). Et n'oubliez pas : `u` annule un remplacement raté.

CHAPITRE 11

Le mode Visuel

Le mode Visuel est le plus intuitif pour les débutants : il ressemble à la sélection à la souris. On surligne une zone, puis on agit dessus.

Entrer en mode Visuel

Touche	Type de sélection
`v`	Visuel **caractère par caractère**
`V`	Visuel **par lignes entières** (très pratique)
`Ctrl`+`v`	Visuel **par bloc** (sélection rectangulaire, en colonnes)

Comment ça marche

Le principe est simple en trois temps :

1. Appuyez sur `v` (ou `V`) pour démarrer la sélection à la position du curseur.
2. **Déplacez-vous** avec les touches habituelles (`h` `j` `k` `l`, `w`, `$`...) : la sélection s'étend.
3. Appuyez sur un opérateur pour agir sur la zone sélectionnée.

Agir sur la sélection

Touche	Effet sur la sélection
`d`	Supprimer (couper) la sélection
`y`	Copier la sélection
`c`	Supprimer la sélection et passer en Insertion
`>`	Indenter (décaler) la sélection vers la droite
`<`	Désindenter vers la gauche
`u` / `U`	Mettre la sélection en minuscules / MAJUSCULES

Le réflexe « V » pour les lignes

Pour traiter plusieurs lignes (les supprimer, les copier, les indenter), `V` est souvent le plus simple : `V` puis `j` `j` `j` sélectionne 4 lignes, puis `d` les coupe ou `>` les indente. Et `Echap` annule la sélection si vous changez d'avis.

CHAPITRE 12

Personnaliser VIM (le fichier .vimrc)

VIM brut est austère. Quelques réglages dans un petit fichier de configuration le rendent nettement plus confortable et lisible, dès aujourd'hui.

Qu'est-ce que le .vimrc ?

Le **.vimrc** est un fichier texte de configuration, lu par VIM à chaque démarrage. On y inscrit ses préférences (affichage, comportement, raccourcis). Il se trouve dans votre dossier personnel :

- Sur **Linux / macOS** : `~/.vimrc`
- Sur **Windows** : `_vimrc` dans le dossier utilisateur

Pour le créer ou l'ouvrir, lancez simplement : `vim ~/.vimrc`

Un .vimrc de départ, commenté

Copiez ces lignes pour une base saine et confortable. Tout ce qui suit un `"` (guillemet) est un commentaire ignoré par VIM.

```
" ---- Affichage ----
set number            " affiche les numéros de ligne
set relativenumber    " numéros relatifs (utile pour 5j, 3k...)
syntax on             " coloration syntaxique du code
set cursorline        " surligne la ligne courante

" ---- Recherche ----
set ignorecase        " recherche insensible à la casse...
set smartcase         " ...sauf si on tape une majuscule
set incsearch         " surligne au fur et à mesure de la frappe
set hlsearch          " surligne tous les résultats trouvés

" ---- Indentation ----
set tabstop=4          " une tabulation = 4 espaces
set shiftwidth=4       " décalage d'indentation = 4 espaces
set expandtab         " remplace les tabulations par des espaces
set autoindent        " garde l'indentation à la ligne suivante

" ---- Confort ----
set wrap              " retour à la ligne visuel des longues lignes
set mouse=a           " autorise la souris (pratique au début)
```

Tester un réglage sans modifier le fichier

Vous pouvez taper n'importe quelle ligne `set` directement en mode Commande pour l'essayer dans la session en cours : par exemple `:set number` puis `Entrée`. Si le réglage vous plaît, ajoutez-le au `.vimrc` pour le rendre permanent.

Avancez doucement

On trouve en ligne des `.vimrc` gigantesques et des centaines de greffons (plugins). En tant que débutant, **résistez à la tentation** de tout copier. Commencez avec la base ci-dessus, comprenez chaque ligne, et ajoutez le reste seulement quand vous en ressentez le besoin.

CHAPITRE 13

Progresser : astuces et bonnes pratiques

Vous avez maintenant toutes les bases. Voici comment transformer ces connaissances en réflexes durables, sans vous décourager.

Comment s'entraîner efficacement

- **Refaites `vimtutor`.** Une deuxième fois après ce cours, tout sera beaucoup plus clair.
- **Utilisez VIM pour de vraies micro-tâches** : éditer un fichier de config, prendre une note. La contrainte réelle ancre l'apprentissage mieux que les exercices abstraits.
- **Imposez-vous `h` `j` `k` `l`** et bannissez les flèches pendant une semaine. C'est le réflexe le plus rentable.
- **Apprenez peu de commandes à la fois.** Maîtrisez-en 3 ou 4 par semaine plutôt que d'essayer de tout retenir d'un coup.

La bonne mentalité

Pensez « phrases », pas « raccourcis »

Quand vous voulez faire quelque chose, formulez-le en langage VIM : « je veux **changer** (c) le texte **entre les guillemets** (i") » → `ci"`. Avec l'habitude, vos doigts traduiront votre intention directement, sans réfléchir.

Pièges fréquents et leurs solutions

Symptôme	Cause probable et solution
« Mon texte déclenche des actions bizarres »	Vous êtes en mode Normal. Faites `i` pour écrire, ou `Echap` puis `u` pour annuler.
« Je n'arrive pas à quitter »	`Echap` puis `:q!` puis `Entrée` (quitte sans enregistrer).
« VIM refuse de quitter (E37) »	Modifications non enregistrées. Faites `:wq` pour enregistrer, ou `:q!` pour abandonner.
« Des surlignages restent après une recherche »	Tapez `:noh` (no highlight) pour les effacer.
« `:` écrit dans le texte »	Vous êtes en Insertion. `Echap` d'abord.

Et après ?

Quand les bases seront automatiques, vous pourrez explorer : les **marques** (sauter à des positions enregistrées), les **macros** (enregistrer et rejouer une suite d'actions), les **fenêtres partagées** (`:split`, `:vsplit`), les **onglets**, et les **greffons** (plugins) pour étendre VIM. Mais rien ne presse : la maîtrise des chapitres précédents couvre déjà l'immense majorité de l'usage quotidien.

ANNEXE A

Aide-mémoire (cheat sheet)

À imprimer ou à garder sous les yeux. Toutes les commandes essentielles du cours, regroupées par usage.

Modes

`Echap`	Revenir en mode Normal (le réflexe de secours)
`i` `a` `o`	Passer en Insertion (avant / après / nouvelle ligne)
`v` `V`	Mode Visuel (caractère / ligne)
`:`	Mode Commande

Fichiers

`:w`	Enregistrer
`:q` / `:q!`	Quitter / quitter sans enregistrer
`:wq` / `ZZ`	Enregistrer et quitter
`:e fichier`	Ouvrir un fichier

Déplacements

`h` `j` `k` `l`	Gauche / bas / haut / droite
`w` `b` `e`	Mot suivant / précédent / fin de mot
`0` `^` `$`	Début / premier caractère / fin de ligne
`gg` `G`	Début / fin du fichier
`nG` ou `:n`	Aller à la ligne n
`f`x	Sauter au caractère x

Éditer

`x`	Supprimer un caractère
`dd` `yy`	Couper / copier une ligne
`dw` `cw`	Supprimer / changer un mot
`p` `P`	Coller après / avant
`r`x	Remplacer un caractère par x
`u` `Ctrl`+`r`	Annuler / refaire
`.`	Répéter la dernière modification

Grammaire & objets

d / y / c + mvt	supprimer / copier / changer une portion
`diw`	supprimer le mot entier
`ci"` `ci(`	changer l'intérieur des " " ou ( )
3dd / 2dw	répéter avec un nombre

Rechercher / remplacer

`/`mot `n` `N`	Chercher, occurrence suivante / précédente
`:%s/a/b/g`	Remplacer a par b dans tout le fichier
`:noh`	Enlever le surlignage de recherche

ANNEXE B

Exercices pratiques

La théorie ne suffit pas : ouvrez VIM et faites réellement ces exercices, dans l'ordre. Chacun renforce un chapitre. Corrigez-vous avec `u` si besoin.

Exercice 1 — Survie (chapitres 3 & 4)

1. Ouvrez un fichier : `vim exo1.txt`
2. Passez en Insertion (`i`) et écrivez trois lignes de votre choix.
3. Revenez en mode Normal (`Echap`), enregistrez et quittez (`:wq`).
4. Rouvrez le fichier, ajoutez une ligne, puis quittez **sans** enregistrer (`:q!`).
5. Vérifiez avec `cat exo1.txt` que la dernière ligne n'a pas été conservée.

Exercice 2 — Navigation (chapitre 5)

1. Dans un fichier d'au moins 20 lignes, allez tout en bas (`G`), puis tout en haut (`gg`).
2. Descendez de 5 lignes d'un coup (`5j`).
3. Allez en fin de ligne (`$`), puis au premier mot (`^`).
4. Avancez de 3 mots (`3w`), puis sautez à la prochaine virgule (`f,`).

Exercice 3 — Édition et grammaire (chapitres 7 & 8)

1. Placez le curseur sur un mot et supprimez-le (`diw`).
2. Copiez une ligne entière (`yy`) et collez-la trois fois (`p` `p` `p`).
3. Écrivez `prix = "0.00"`, placez le curseur entre les guillemets et changez le contenu en une seule commande (`ci"`).
4. Déplacez une ligne plus bas : coupez-la (`dd`), descendez, collez (`p`).

Exercice 4 — Recherche, remplacement, répétition (chapitres 9 & 10)

1. Écrivez plusieurs fois le mot « chat » dans le fichier.
2. Remplacez toutes les occurrences par « chien » avec confirmation (`:%s/chat/chien/gc`).
3. Supprimez un mot avec `dw`, déplacez-vous sur un autre, et répétez avec `.`
4. Annulez les trois dernières actions (`u` `u` `u`), puis rétablissez-en une (`Ctrl`+`r`).

Objectif

Quand vous enchaînez ces quatre exercices sans consulter l'aide-mémoire, vous avez atteint le niveau « débutant autonome » : vous pouvez vous servir de VIM au quotidien. Bravo !

ANNEXE C

Ressources pour aller plus loin

Quelques pistes fiables et gratuites pour continuer votre apprentissage à votre rythme.

Intégré à VIM (rien à installer)

- `vimtutor` — le tutoriel officiel interactif, dans votre langue. À refaire sans modération.
- **L'aide intégrée** — depuis VIM, tapez `:help` puis `Entrée`. Pour une commande précise : `:help dw`, `:help :s`, etc. C'est la documentation la plus complète qui soit.

Pour s'entraîner de façon ludique

- **VIM Adventures** — un jeu en ligne où l'on apprend les déplacements et commandes en jouant.
- **Open VIM** — un bac à sable interactif dans le navigateur pour tester les commandes sans rien installer.

Documentation et références

- **Site officiel** — `vim.org` (téléchargement, documentation, communauté).
- **VIM Cheat Sheet** — `vim.rtorr.com`, une fiche de référence en ligne très complète.
- **Neovim** — `neovim.io` si vous souhaitez explorer la variante moderne (les commandes de ce cours s'y appliquent).

Le mot de la fin

Apprendre VIM, c'est comme apprendre à faire du vélo : maladroit au début, puis totalement naturel. Vous avez désormais tout ce qu'il faut pour démarrer. Ouvrez un fichier, et laissez vos doigts apprendre. Bon voyage dans VIM !