

mode puissant qui permet de se déplacer dans le texte, de supprimer une ligne, copier-coller du texte, rejoindre une ligne précise, annuler ses actions, etc. Chaque action peut être déclenchée en appuyant sur une touche du clavier (par exemple, on appuie sur `u` pour annuler la dernière action).

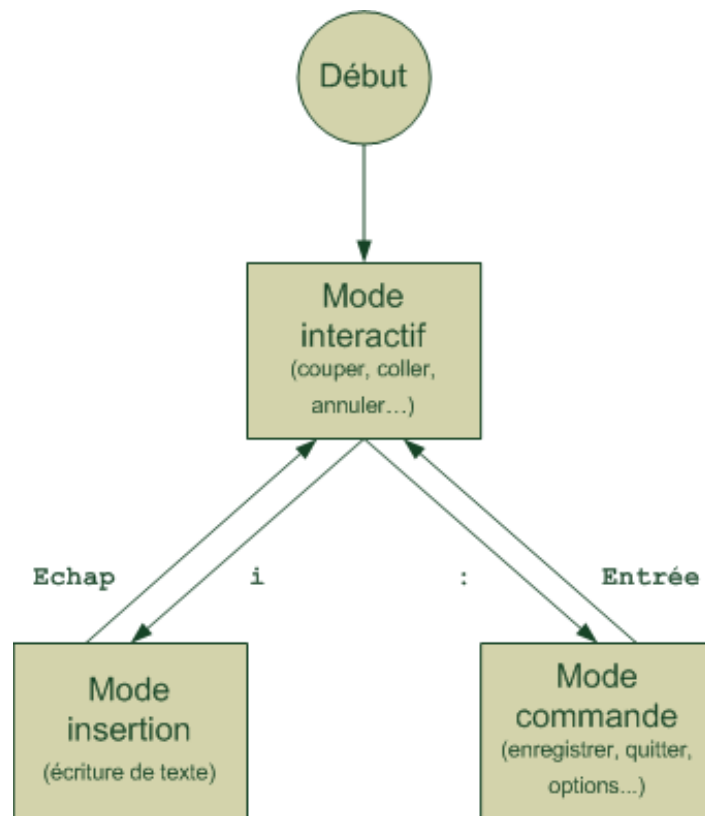
- **Mode insertion** : celui-là, c'est celui que vous connaissez ! Vous tapez du texte et ce dernier s'insère à l'endroit où se trouve le curseur.

Pour entrer dans ce mode, il existe plusieurs possibilités. L'une des plus courantes est d'appuyer sur la touche `i` (*insertion*). Pour en sortir, il faut appuyer sur la touche `Echap`.

- **Mode commande** : ce mode permet de lancer des commandes telles que « quitter », « enregistrer », etc. Vous pouvez aussi l'utiliser pour activer des options de Vim (comme la coloration syntaxique, l'affichage du numéro des lignes...). Vous pouvez même envoyer des commandes au shell (la console) telles que `ls`, `locate`, `cp`, etc.

Pour activer ce mode, vous devez être en mode interactif et appuyer sur la touche deux points « `:` ». Vous validerez la commande avec la touche `Entrée` et reviendrez alors au mode interactif.

Je résume. Vim possède trois modes (figure suivante) : interactif, insertion et commande. Vous démarrez en mode interactif. Le seul mode que vous connaissez et qui ne sera pas nouveau pour vous est le mode insertion. Les deux autres modes (interactif et commande) vont quelque peu vous surprendre.



Pourquoi avoir intégré dans un éditeur de texte autant de modes ayant l'air si complexes ? Pourquoi n'y a-t-il pas de menus ? Et pourquoi ne pas utiliser plutôt un éditeur de texte graphique ? C'est quand même plus simple avec une souris !

Cela fait beaucoup de questions dites donc. 😊

Je vais essayer de vous répondre simplement et, dans un premier temps, il va falloir que vous me croyiez sur parole : si des gens se sont amusés à créer tous ces « modes » et tous ces raccourcis clavier, ce n'est pas juste pour le plaisir tordu de faire des choses compliquées.

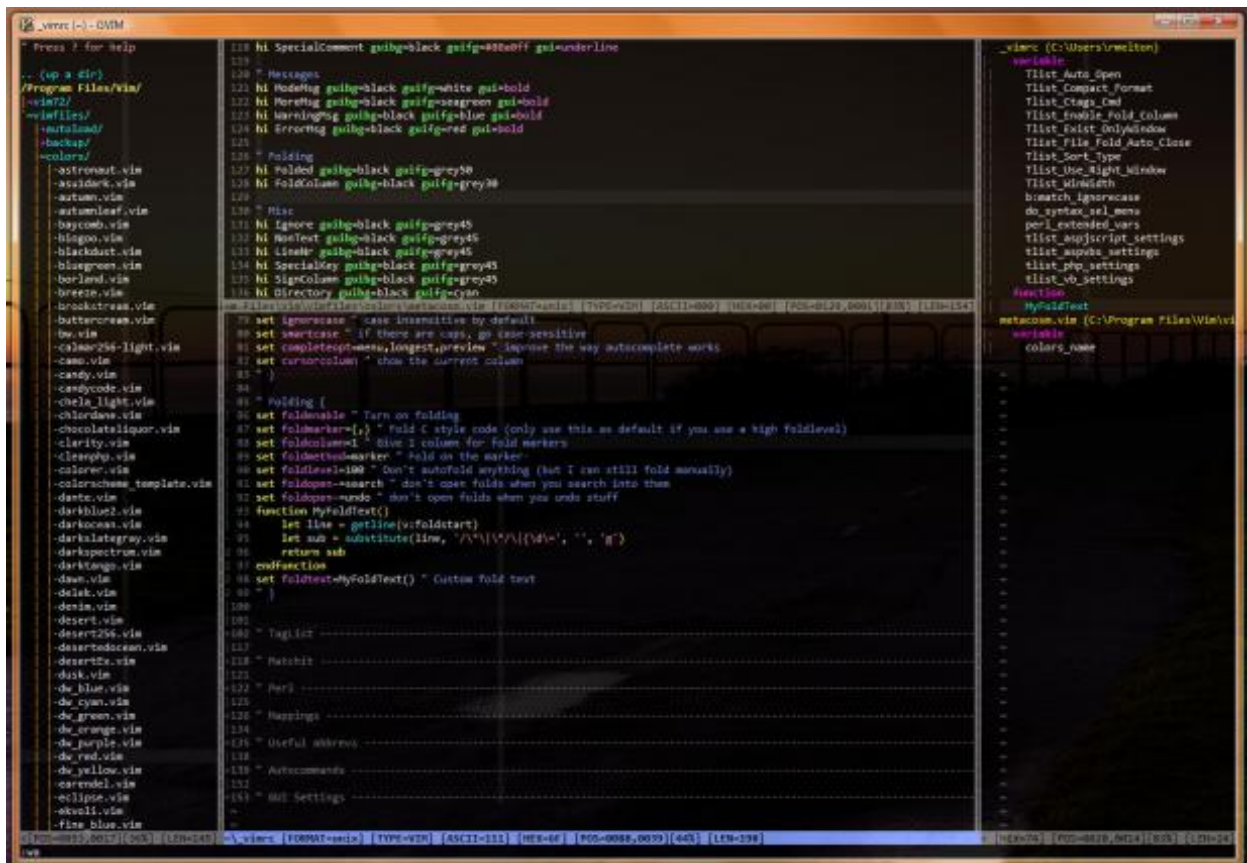
En fait, vous allez rapidement vous rendre compte que **vous pouvez faire des choses que vous ne soupçonniez pas réalisables avec un éditeur de texte** : supprimer le mot actuel, couper le texte du curseur jusqu'à la fin de la ligne, coller quatre fois le texte qui se trouve dans le presse-papier, sauter à la ligne n° 453, sauter à la dernière ligne, etc.

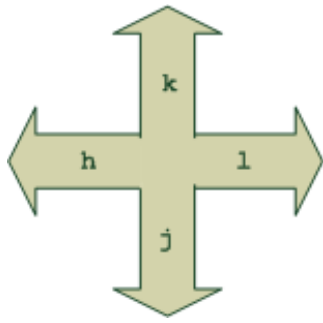
Toutes ces choses-là se font au clavier et, pour la plupart d'entre elles, vous devrez retenir par cœur quelle touche correspond à quelle action. C'est un peu contraignant au départ, mais imaginez que c'est comme apprendre à taper des dix doigts au clavier comme un dactylo : au début, c'est difficile ; vous avez l'impression de ramer, d'aller moins vite qu'avant, mais petit à petit vous gagnez en productivité, vous allez de plus en plus vite et vous finissez par vous demander comment vous avez pu rester autant

de temps sans connaître tout ça. 😊

Et pour ceux qui voudraient une interface graphique, sachez que Vim a été porté en interface graphique sous le nom « gVim » (ou vim-gnome selon les versions). Vous pouvez donc l'installer (même si vous utilisez KDE, cela fonctionnera) et le lancer : le fonctionnement est identique à celui du Vim de la console. Il est même disponible en version Windows (figure suivante)... si ce n'est pas beau, ça !

Par défaut, cette fenêtre affiche des menus et une barre d'outils, comme un éditeur de texte classique. Un habitué du Vim console aura bien entendu plutôt tendance à utiliser les raccourcis clavier, qui permettent de gagner du temps.





QUOIII ? C'est le comble ! On ne peut même pas utiliser les flèches du clavier pour se déplacer ?!

Si si, vous pouvez également les utiliser : vous n'avez qu'à essayer pour voir. D'ailleurs, en mode insertion, c'est la seule chose qui fonctionne.

0 et \$: se déplacer en début et fin de ligne

Pour placer le curseur au tout début de la ligne, appuyez sur 0 en mode interactif. La touche *Origine* que vous avez peut-être l'habitude d'utiliser fonctionne aussi. Cependant, reprenez plutôt qu'il faut utiliser 0, ça vous sera utile par la suite.

De même, pour se rendre en fin de ligne, appuyez sur la touche \$.

Là encore, la touche *Fin* fonctionne elle aussi, mais essayez de prendre l'habitude d'utiliser \$; ce sera payant, vous allez voir.

w : se déplacer de mot en mot

Avec w, vous pouvez vous déplacer de mot en mot dans le fichier. C'est un autre moyen, parfois plus efficace et plus rapide, pour se déplacer au sein d'une ligne du fichier.

:w : enregistrer le fichier

Pour enregistrer votre fichier, vous devez être au préalable en mode interactif (appuyez sur *Echap* pour vous en assurer).

Appuyez ensuite sur la touche deux points « : » pour passer en mode commande, puis tapez w (*write*) suivi du nom du fichier. La commande doit s'afficher en bas.

Dans mon cas, j'ai donc tapé :w monfichier (figure suivante). Appuyez ensuite sur la touche *Entrée* pour valider. Le bas de l'écran doit indiquer que le fichier a été écrit (*written*) :

```
"monfichier" [New] 4L, 185C written                                4,101-  
98          All
```

Notez que j'aurais tout aussi bien pu donner une extension .txt à mon fichier.

x : effacer des lettres

Nous avons vu le strict minimum de ce qu'il faut connaître pour se débrouiller dans Vim. Si cela n'a rien de difficile, il faut bien avouer que c'est tout de même perturbant. Prenez donc le temps de vous y habituer.

À présent, allons un peu plus loin. Vous allez d'ailleurs commencer à trouver Vim pratique (et parfois même étonnant). Nous allons effectuer la majorité de ces actions en mode interactif : appuyez sur la touche Echap si vous n'y êtes pas déjà.

Placez le curseur sur une lettre en mode interactif puis appuyez sur x pour l'effacer. Cela revient à appuyer sur Suppr en mode insertion.

On peut aller plus loin et effacer plusieurs lettres d'un coup. Pour cela, utilisez la formule suivante :

(nombre)x

Par exemple, si vous tapez 4x (4 puis x), vous supprimerez les quatre prochaines lettres en partant du curseur.

Vous devez taper 4 puis x. Ne vous étonnez pas si rien ne s'affiche à l'écran lorsque vous tapez 4 : c'est normal. Écrivez la commande jusqu'au bout, cela fonctionnera.

d : effacer des mots, des lignes...

De la même manière, on utilise la touche d pour supprimer des mots et des lignes.

Commençons par supprimer une ou plusieurs lignes.

dd : supprimer une ligne

Appuyez deux fois sur d (dd) pour supprimer toute la ligne sur laquelle se trouve le curseur.

Mieux : vous pouvez faire précéder cette instruction d'un nombre de lignes à supprimer. Par exemple, si vous tapez 2dd, vous supprimerez deux lignes d'un coup.

Encore une fois, ne vous étonnez pas si juste après avoir tapé 2 rien ne s'affiche à l'écran. L'information est gardée en mémoire par Vim, mais l'action ne sera vraiment exécutée que lorsque vous aurez tapé entièrement 2dd.

Note importante : la ligne ainsi supprimée est en fait « coupée » et placée en mémoire. Elle peut être collée, comme on le verra plus loin, avec la touche p.

dw : supprimer un mot

Placez le curseur sur la première lettre d'un mot. Tapez ensuite dw (*delete word*) : cela supprime le mot complet !

Si le curseur est positionné au milieu du mot, vous ne supprimerez que les prochains caractères de celui-ci (jusqu'à l'espace qui suit).

Vous pouvez aussi supprimer les trois prochains mots en tapant 3dw. Notez que le 3 peut être placé entre le d et le w ; cela revient au même : d3w (qui peut se lire « *delete 3 words* »).

d0 et d\$: supprimer le début ou la fin de la ligne

Vous souvenez-vous de 0 et de \$? Je vous avais demandé de les utiliser à la place des touches Origine et Fin car nous en aurions à nouveau besoin par la suite. Le moment est venu de s'en resservir.

- En tapant d0, vous supprimez du curseur jusqu'au début de la ligne.
- En tapant d\$, vous supprimez du curseur jusqu'à la fin de la ligne.

Pratique !

yy : copier une ligne en mémoire

yy copie la ligne actuelle en mémoire.

Cela fonctionne comme dd, qui lui la « coupe ». Vous pouvez aussi utiliser yw pour copier un mot, y\$ pour copier du curseur jusqu'à la fin de la ligne, etc.

p : coller

Si vous avez « coupé » du texte avec dd ou copié du texte avec yy (ou un de leurs équivalents) vous pouvez ensuite le coller avec la touche p.

Attention, retenez bien ceci : si vous avez copié une ligne en mémoire et que vous appuyez sur p, elle sera collée sur **la ligne située après le curseur**.

On est parfois surpris de voir où se colle le texte ; prenez donc le temps de vous y habituer.

Vous pouvez aussi coller plusieurs fois un texte en faisant précéder le p d'un nombre. Par exemple, 8p collera huit fois le texte en mémoire.

Si je place mon curseur sur une ligne, que je tape yy puis 8p, je la collerai donc huit fois (figure suivante) !

```
Salut les Zéros !
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
Mine de rien, une fois qu'on a basculé en mode insertion, écrire du texte n'est
pas compliqué. ;o)
~
~
~
~
~
~
~
~
~
~
8 more lines                               4,1           All
```

r : remplacer une lettre

Si vous avez fait une faute sur une lettre seulement, vous pouvez passer en mode remplacement.

Placez le curseur sur la lettre à remplacer. Tapez « r » suivi de la lettre que vous voulez mettre à la place. Par exemple, rs remplace la lettre actuelle par un « s ».

Si vous utilisez un « R » majuscule, vous basculerez cette fois dans le mode remplacement : vous pourrez alors remplacer plusieurs lettres à la fois. Vous pouvez par exemple écrire « Rbonjour » pour remplacer les caractères par « bonjour ».

Pour revenir au mode interactif normal, appuyez sur Echap.

u : annuler les modifications

Pour annuler vos dernière modifications, appuyez sur u (*undo*). Si vous souhaitez annuler vos quatre dernières modifications, appuyez sur 4u.

Vous commencez à connaître la formule, c'est toujours la même. 😊

Pour répéter un changement (= annuler une annulation), appuyez sur `Ctrl + R`.

G : sauter à la ligne n° X

Toutes les lignes d'un fichier possèdent un numéro. La numérotation commence à 1.

Regardez bien en bas à droite de Vim, vous devriez voir quelque chose comme 4,3. 4 correspond au numéro de la ligne sur laquelle se trouve le curseur, et 3 au numéro de la colonne (3e lettre de la ligne).

Vous pouvez par exemple directement sauter à la ligne n° 7 en tapant `7G` (attention, c'est un « G » majuscule, donc pensez à laisser la touche `Maj` appuyée).

Pour sauter à la dernière ligne, tapez simplement `G`.

Pour revenir à la première ligne, tapez `gg`.

/ : rechercher un mot

Nous avons vu l'essentiel des commandes les plus courantes. Nous allons maintenant découvrir une série de commandes un peu plus complexes parmi lesquelles la fusion de fichiers, la recherche, le remplacement, le découpage de l'écran (*split*), etc.

Toutes ces commandes se lancent depuis le mode interactif.

Si vous tapez /, vous passez en mode recherche. Le curseur se place en bas de l'écran (vous indiquant que vous êtes passés en mode commande).

Écrivez ensuite le mot que vous recherchez, par exemple « remplir » : /remplir. Tapez ensuite sur Entrée pour valider.

Le curseur se place alors sur la prochaine occurrence de « remplir » dans le fichier. Pour passer à la prochaine occurrence du mot, plus bas dans le fichier (s'il apparaît plusieurs fois), appuyez sur n. Pour rechercher en arrière, appuyez sur N (Maj + n).

Si vous souhaitez dès le départ lancer une recherche qui remonte vers le début du fichier, utilisez ? au lieu de / pour lancer la recherche ; le fonctionnement reste le même.

:s : rechercher et remplacer du texte

Pour rechercher et remplacer du texte, c'est un peu plus compliqué. Il y a en effet plusieurs façons d'effectuer le remplacement.

La plus simple façon d'effectuer une recherche consiste à taper :s/ancien/nouveau pour rechercher « ancien » et le remplacer par « nouveau ». Le problème... c'est que cela ne remplacera que la première occurrence d'« ancien » par « nouveau ».

Voici toutes les variantes à connaître :

- :s/ancien/nouveau : remplace la première occurrence de la ligne où se trouve le curseur ;
- :s/ancien/nouveau/g : remplace toutes les occurrences de la ligne où se trouve le curseur ;
- :#, #s/ancien/nouveau/g : remplace toutes les occurrences dans les lignes n° # à # du fichier ;

- `:%s/ancien/nouveau/g` : remplace toutes les occurrences dans tout le fichier. C'est peut-être ce que vous utiliserez le plus fréquemment.

:r : fusion de fichiers

Avec `:r`, vous pouvez insérer un fichier à la position du curseur. Vous devez indiquer le nom du fichier à insérer, par exemple : `:r autrefichier`.

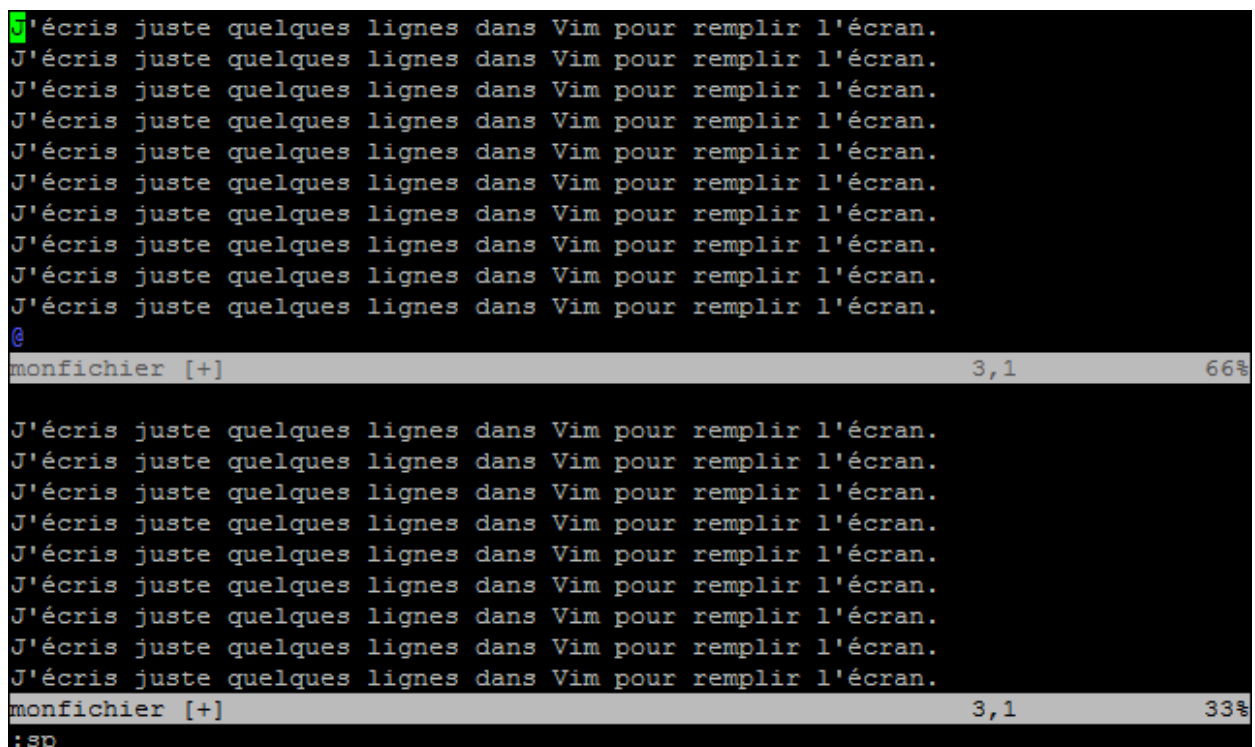
L'autocomplétion avec Tab fonctionne là aussi, donc pas besoin d'écrire le nom du fichier en entier !

Le découpage d'écran (split)

Vim possède une fonctionnalité pratique : il permet de découper l'écran et d'ouvrir plusieurs fichiers.

:sp : découper l'écran horizontalement

Le plus simple pour commencer est de découper l'écran horizontalement. Tapez la commande `:sp` pour scinder l'écran en deux, comme sur la figure suivante.



```
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
@
monfichier [+] 3,1 66%
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
monfichier [+] 3,1 33%
:sp
```

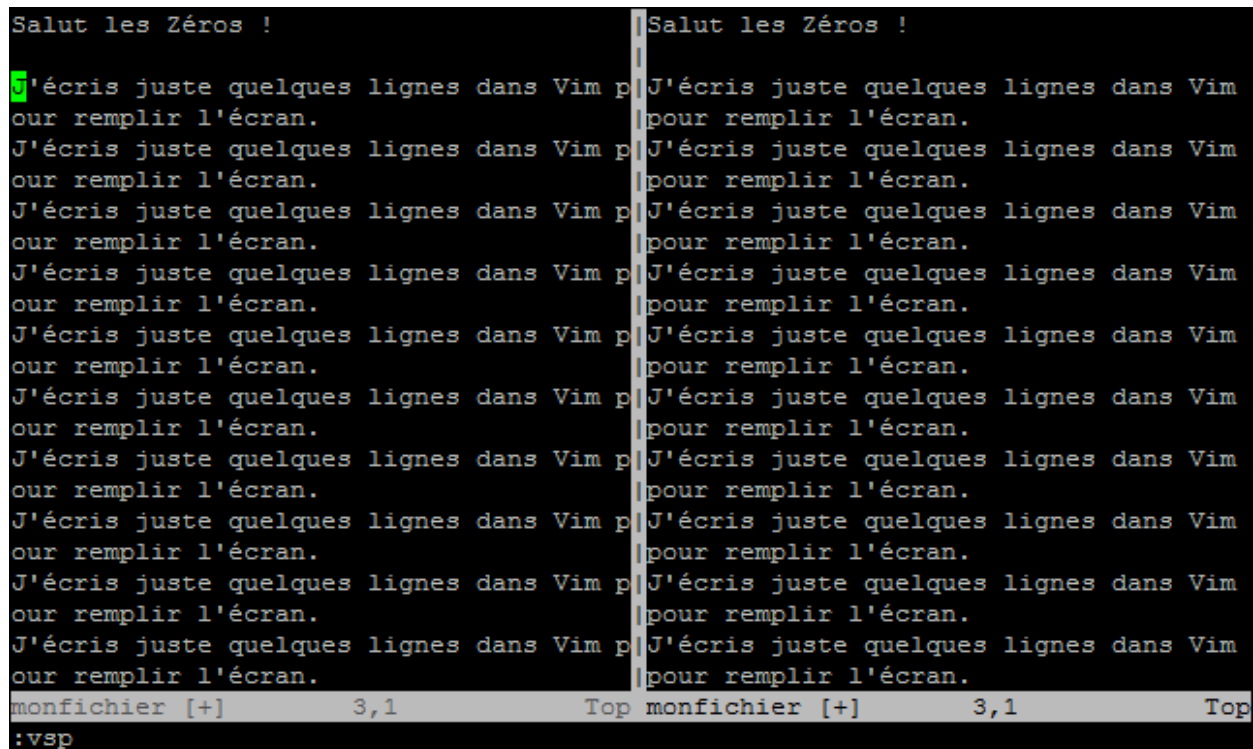
Le fichier est ouvert une seconde fois (ce qui vous permet de voir deux endroits différents du fichier à la fois) mais il est bien entendu possible d'ouvrir deux fichiers différents. Pour cela, ajoutez le nom du fichier à ouvrir à la suite de la commande : `:sp autrefichier`. Bonne nouvelle : l'autocomplétion à l'aide de la touche Tab

fonctionne aussi dans Vim !

Vous pouvez cette fois-ci taper à nouveau :sp pour scinder l'écran en trois et ainsi de suite, mais gare à la lisibilité !

:vsp : découper l'écran verticalement

Si le découpage horizontal par défaut ne vous convient pas, sachez que vous pouvez aussi effectuer un découpage vertical avec :vsp (figure suivante).



```
Salut les Zéros !
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
monfichier [+] 3,1 Top
:vsp
```

Il est bien entendu possible de répéter plusieurs fois la commande et même de combiner des découpages verticaux et horizontaux.

Les principaux raccourcis en écran splitté

Chaque morceau de l'écran (correspondant à un fichier) est appelé **viewport**.

Voici une liste de raccourcis pratiques que vous pouvez utiliser lorsque l'écran est splitté (scindé).

- **Ctrl + w** puis **Ctrl + w** : navigue de viewport en viewport. Répétez l'opération plusieurs fois pour accéder au viewport désiré.
- **Ctrl + w** puis **j** : déplace le curseur pour aller au viewport juste en dessous. La même chose fonctionne avec les touches **h**, **k** et **l** que l'on utilise traditionnellement pour se déplacer dans Vim.

- `Ctrl + w` puis `+` : agrandit le viewport actuel.
- `Ctrl + w` puis `-` : réduit le viewport actuel.
- `Ctrl + w` puis `=` : égalise à nouveau la taille des viewports.
- `Ctrl + w` puis `r` : échange la position des viewports. Fonctionne aussi avec « R » majuscule pour échanger en sens inverse.
- `Ctrl + w` puis `q` : ferme le viewport actuel.

Voilà qui devrait vous permettre de faire ce que vous voulez en écran splitté.

: ! : lancer une commande externe

Il est possible d'écrire des commandes traditionnelles du shell directement dans Vim. Pour cela, commencez par taper `: !` suivi du nom de la commande.

Essayez par exemple de taper `: !ls`. Vous afficherez alors le contenu du dossier dans lequel vous vous trouvez !

Cette fonctionnalité est bien pratique pour effectuer quelques actions sans avoir à quitter Vim.

Le fonctionnement des options

Vim peut être personnalisé de deux façons différentes :

- En activant ou désactivant des options. [La documentation complète des options](#) est disponible en ligne.
- En installant des plugins. Voyez [la page officielle des plugins les plus téléchargés de Vim](#).

Nous n'allons pas passer en revue les plugins, mais il y a un certain nombre d'options intéressantes qui valent le coup d'être activées.

Les options peuvent être activées après le démarrage de Vim en lançant des commandes. Cependant, ces options seront « oubliées » dès que vous quitterez le logiciel.

Si vous voulez que les options soient activées à chaque démarrage de Vim, il faut créer un fichier de configuration `.vimrc` dans votre répertoire personnel.

Activer des options en mode commande

La première méthode consiste à activer l'option en mode commande. Une fois Vim ouvert, pour activer l'option nommée « option », tapez :

```
:set option
```

Pour la désactiver, tapez :

```
:set nooption
```

Il faut donc ajouter le préfixe `no` devant le nom de l'option pour la désactiver.

Certaines options doivent être précisées avec une valeur, comme ceci :

```
:set option=valeur
```

Pour connaître l'état d'une option :

```
:set option?
```

Activer des options dans un fichier de configuration

C'est à mon avis la meilleure façon de procéder. Commencez par copier un fichier de configuration déjà commenté qui vous servira d'exemple : il y en a un dans `/etc/vim` qui s'appelle `vimrc`.

Copiez-le dans votre répertoire personnel en le faisant précéder d'un point (pour que ce soit un fichier caché) :

```
$ cp /etc/vim/vimrc ~/.vimrc
```

Ouvrez maintenant ce fichier... avec Vim, bien sûr.

```
$ vim .vimrc
```

Le début du fichier ressemble à ceci :

```
" All system-wide defaults are set in $VIMRUNTIME/debian.vim (usually just
" /usr/share/vim/vimcurrent/debian.vim) and sourced by the call to :runtime
" you can find below.  If you wish to change any of those settings, you should
"
" do it in this file (/etc/vim/vimrc), since debian.vim will be overwritten
" everytime an upgrade of the vim packages is performed.  It is recommended to
"
" make changes after sourcing debian.vim since it alters the value of the
" 'compatible' option.
"
" This line should not be removed as it ensures that various options are
" properly set to work with the Vim-related packages available in Debian.
runtime! debian.vim
"
" Uncomment the next line to make Vim more Vi-compatible
" NOTE: debian.vim sets 'nocompatible'.  Setting 'compatible' changes numerous
" options, so any other options should be set AFTER setting 'compatible'.
"set compatible
"
" Vim5 and later versions support syntax highlighting. Uncommenting the next
" line enables syntax highlighting by default.
"syntax on
"
" If using a dark background within the editing area and syntax highlighting
" turn on this option as well
```

Les lignes commençant par « " » sont des commentaires. Je vous recommande de les lire, ils fournissent des informations utiles.

Passons maintenant à l'activation de quelques commandes bien utiles. Je vous recommande de travailler comme moi, avec le fichier de configuration `.vimrc`, et d'activer les options qui vous plaisent en décommentant les lignes concernées.

Pour cela, la meilleure façon de procéder est de se mettre en mode interactif, de se déplacer avec `hjkl` et d'appuyer sur `x` lorsque le curseur est sur un guillemet pour le supprimer et activer ainsi l'option.

syntax : activer la coloration syntaxique

Il s'agit clairement de la première option à activer : la coloration syntaxique. En fonction du type de fichier que vous ouvrez, Vim colorera le texte.

Vim supporte un très très grand nombre de langages de programmation : C, C++, Python, Java, Ruby, Bash, Perl, etc.

Activez donc l'option :

```
syntax on
```


Lors d'une recherche, si vous souhaitez que Vim ne fasse pas la différence entre les majuscules et les minuscules, activez cette option :

```
set ignorecase
```

mouse : activer le support de la souris

Eh oui ! Même en mode console, il est possible d'utiliser la souris.

Commencez par activer le support de cette dernière :

```
set mouse=a
```

Désormais, vous pourrez cliquer avec la souris sur une lettre pour y déplacer le curseur directement. Vous pourrez également utiliser la molette de la souris pour vous déplacer dans le fichier.

Il vous sera également possible de sélectionner du texte à l'aide de la souris. Vous passerez alors en mode visuel.

Dans ce mode, vous pouvez supprimer le texte sélectionné (avec `x`, comme d'habitude), mais aussi mettre le texte tout en majuscules (`U`), minuscules (`u`), etc.

En résumé

- Vim est un éditeur de texte très puissant en console et qui offre plus de possibilités que Nano, que nous avons découvert plus tôt dans cet ouvrage. Son grand concurrent est Emacs.
- Dans Vim, il existe trois modes : interactif, insertion et commande.
- Le mode par défaut est le mode interactif. Il faut appuyer sur la touche `i` pour insérer du texte et sur la touche `Echap` pour revenir au mode interactif.
- On peut lancer des commandes en appuyant sur la touche deux points « `:` » depuis le mode interactif. Par exemple, `:w` enregistre le fichier, `:q` quitte Vim et `:wq` effectue les deux à la fois.
- Il existe de nombreux raccourcis à connaître pour bien utiliser Vim ; il faut prendre le temps de les apprendre pour exploiter pleinement le logiciel.
- On peut modifier le fichier `.vimrc` pour activer certaines options de Vim, comme la coloration automatique du code.