Reprenez le contrôle à l'aide de Linux !

30 heures Facile

Mis à jour le 29/06/2021

Le contenu de ce cours n'est plus à jour

Nous avons archivé ce cours et n'actualiserons plus son contenu.

Accédez au contenu le plus récent en découvrant ce cours :

SYSTÈMES & RÉSEAUX

Initiez-vous à Linux

Easy 8 heures

Dans ce cours débutant, découvrez Linux : un système d'exploitation gratuit et fascinant qui vous donnera un contrôle sans précédent sur votre ordinateur ! Créé par des passionnés d'informatique, Linux est un vecteur important de la philosophie du libre et l'alternative parfaite à Windows ou macOS.

VOIR LE NOUVEAU COURS

Vim : l'éditeur de texte du programmeur

Dans cette dernière partie, nous allons réunir toutes les connaissances que nous avons acquises concernant les commandes utilisées sous Linux. Nous allons les combiner et créer ce que l'on appelle **des scripts shell**.

Le scripting shell est un minilangage de programmation intégré à tous les systèmes Linux et qui vous permet d'automatiser des tâches répétitives. Il s'agit d'un élément très puissant du système que vous devez absolument connaître.

Toutefois, pour programmer, il va vous falloir utiliser un éditeur de texte digne de ce nom. Certes, vous connaissez déjà Nano mais comme je vous l'ai dit ce dernier est très basique. Nous l'avons utilisé au départ pour simplifier, mais il est temps à présent de passer à quelque chose de plus complet et de plus puissant : Vim (prononcez « *Vi aille ème* »).

Installer Vim

Sous Linux, deux puissants éditeurs de texte en console sont à connaître.

- **Vim** : il s'agit d'une version améliorée de l'un des plus anciens éditeurs en console : « Vi » (prononcez les lettres en anglais « *Vi aille* »).

Vim (*VI iMproved*, version améliorée de Vi) est largement répandu et généralement disponible par défaut sur la plupart des OS basés sur Unix, comme Linux.

Emacs : développé par Richard Stallman, le fondateur du projet GNU dont je vous ai parlé au début du livre, cet éditeur concurrent a lui aussi bien des atouts. On le retrouve plus spécifiquement sous Linux mais il est rarement installé par défaut (un petit `apt-get` suffit, toutefois). Il peut être complété par toute une série de plugins qui lui permettent de faire navigateur web, lecteur audio... Bref, c'est un peu un outil à tout faire.

Sachez qu'il est courant que les gens adoptent et défendent bec et ongles l'un ou l'autre de ces éditeurs. Choisir un éditeur de texte sous Linux, c'est en fait un peu comme choisir une religion (oui, je sais : ils sont fous, ces Linuxiens !).

Hou ! là, c'est important alors ! Lequel choisir ?

En fait, rien ne vous empêche d'apprendre à utiliser les deux. Toutefois ces logiciels sont tellement complets qu'il vous faudra du temps pour vous habituer à chacun d'eux.
Dans la pratique, on prend l'habitude d'en choisir un et de s'y tenir : il est donc rare de voir quelqu'un naviguer entre les deux.

Vim ou Emacs ? Emacs ou Vim ?

Tout cela ne répond pas à votre question, je sais. Mais ne comptez pas sur moi pour vous dire « Utilisez celui-là, il est mieux » : des milliers de *trolleurs* le font mieux que moi sur tous les forums du monde. Et je pourrais m'attirer les foudres divines des adorateurs de l'un ou l'autre éditeur si je m'y risquais.

D'ailleurs, vous devriez vous mettre en tête dès maintenant qu'il n'y en a pas un qui soit nul et l'autre génial ; ce sont juste deux conceptions un peu différentes de ce que doit être un éditeur de texte.

Le meilleur conseil que je puisse vous donner est le suivant : **choisissez d'utiliser le même éditeur que votre ami pro de Linux ou votre collègue de bureau**. L'idéal est d'avoir quelqu'un à proximité qui peut régulièrement vous conseiller. Croyez-moi, s'il est bien un conseil qui soit important dans ce chapitre, c'est celui-là.

Et toi, ton éditeur, c'est quoi ?

Je craignais cette question mais il fallait bien qu'elle soit posée un jour...
Pour ma part, je n'ai jamais eu l'occasion de prendre le temps d'apprendre à utiliser Emacs. Le professeur qui m'a initié à Linux était un habitué de Vim (mais il n'a jamais dit du mal d'Emacs, je le jure !).

Je suis donc à mon tour un habitué de Vim et c'est lui que je vous présenterai dans ce livre.

Installer et lancer Vim

Sur la plupart des distributions Linux, Vim est en général installé par défaut. J'ai bien dit **en général**. En effet, rien n'assure que Vim soit installé par défaut sur votre distribution ; après tout, c'est elle qui choisit les programmes initialement installés.

Sous Ubuntu, il faut savoir que ce n'est pas Vim qui est installé mais **Vim-tiny**, une version allégée. Personnellement, elle ne me convient pas ; de plus, elle est limitée en possibilités. Je vous invite donc à installer le vrai Vim complet en tapant :

```
sudo apt-get install vim
```

Vous pourrez alors lancer le logiciel en tapant la commande `vim` .
La commande `vi` fonctionne aussi mais il est recommandé de taper plutôt `vim` .

Vimtutor : le programme qui vous apprend à utiliser Vim !

Pour les nouveaux utilisateurs, Vim intègre un véritable petit tutoriel !
Ce programme peut être lancé en tapant :

```
vimtutor
```

Si vous ne l'avez pas, installez le paquet `vim-common` ... mais normalement il devrait déjà être présent sur votre distribution.

En fait, Vimtutor lance simplement Vim en ouvrant un fichier d'aide prédéfini. Cette introduction à Vim est d'ailleurs en français et accessible à tout le monde, aussi je vous invite à l'essayer et à la lire en complément de ce qui suit.

Petit aperçu :

```
===============================================================================
= B i e n v e n u e  dans le  T u t o r i e l  de  V I M  -  Version 1.5.fr.2
===============================================================================

     Vim est un éditeur très puissant qui a trop de commandes pour pouvoir
     toutes les expliquer dans un cours comme celui-ci, qui est conçu pour en
     décrire suffisamment afin de vous permettre d'utiliser simplement Vim.

     Le temps requis pour suivre ce cours est d'environ 25 à 30 minutes, selon
     le temps que vous passerez à expérimenter.  Les commandes utilisées dans
     les leçons modifieront le texte. Faites une copie de ce fichier afin de
     vous entraîner dessus (si vous avez lancé "vimtutor" ceci est déjà une
     copie).

     Il est important de garder en tête que ce cours est conçu pour apprendre
     par la pratique.  Cela signifie que vous devez exécuter les commandes
     pour les apprendre correctement.  Si vous vous contentez de lire le
     texte, vous oublierez les commandes !

     Maintenant, vérifiez que votre clavier n'est PAS verouillé en majuscules,
     et appuyez la touche  j  le nombre de fois suffisant pour que la leçon
     1.1 remplisse complètement l'écran.
```

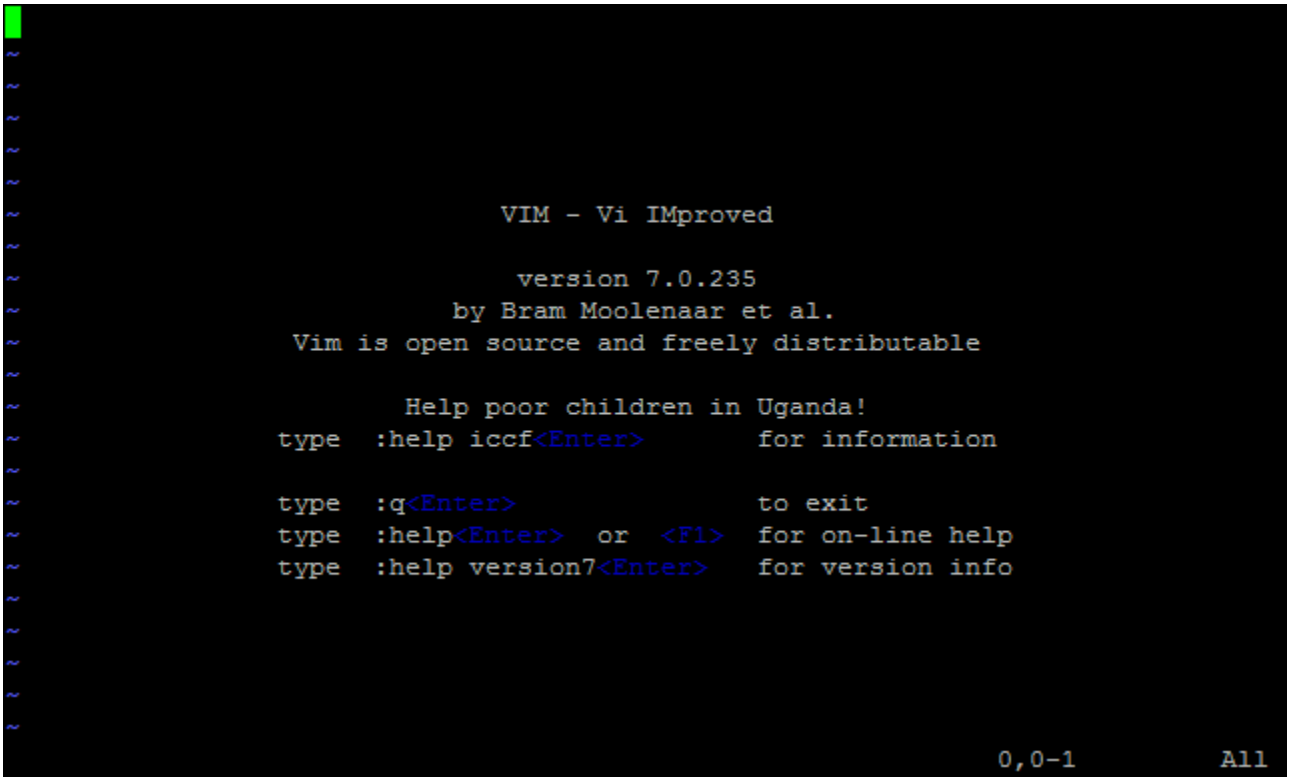
Il faut compter en général une bonne demi-heure pour suivre le Vimtutor. Cela vous fait une bonne petite introduction au logiciel, mais gardez bien entendu à l'esprit que les possibilités sont bien plus grandes et que vous n'aurez pas tout vu à l'issue de sa lecture.

Les modes d'édition de Vim

Commencez par lancer Vim. Comme je vous l'ai dit plus tôt, il suffit pour cela de taper la commande suivante :

```
vim
```

Vim s'ouvre alors (figure suivante).



Vim est un programme un peu surprenant qui ne s'utilise pas comme la plupart des éditeurs de texte que vous connaissez. Il m'a fallu un peu de temps pour m'y habituer et il vous en faudra aussi, mais le jeu en vaut la chandelle.

Si on ne vous explique rien, vous risquez d'être un peu perdus. Pire, vous aurez même du mal à sortir de Vim (ne riez pas, ça m'est arrivé la première fois !).

Voilà ce qu'il faut savoir... et qu'on aurait dû m'expliquer dès le départ, d'ailleurs. Vim possède trois modes de travail différents.

- **Mode interactif** : c'est le mode par défaut par lequel vous commencez. En lançant Vim, vous êtes donc en mode interactif.

Dans ce mode, vous ne pouvez pas écrire de texte (oui, je sais, il s'agit d'un comble pour un éditeur de texte !). N'essayez donc pas d'appuyer sur des lettres au hasard car vous risqueriez de faire n'importe quoi !Le mode interactif est un 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).

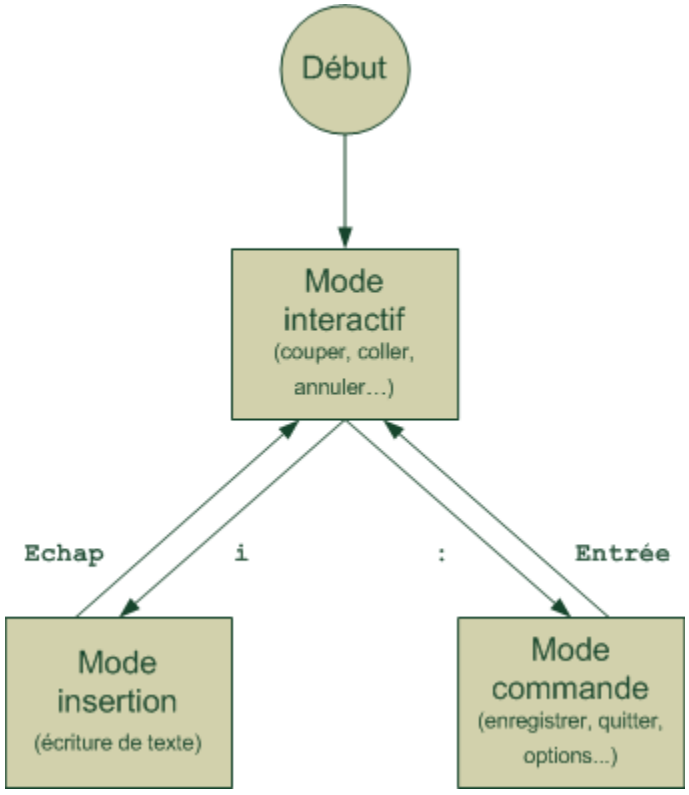
Mode insertion : celui-là, c'est celui que vous connaissez ! Vous tapez du texte et ce dernier s'insère à l'endroit où se trouve le curseur.

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 !

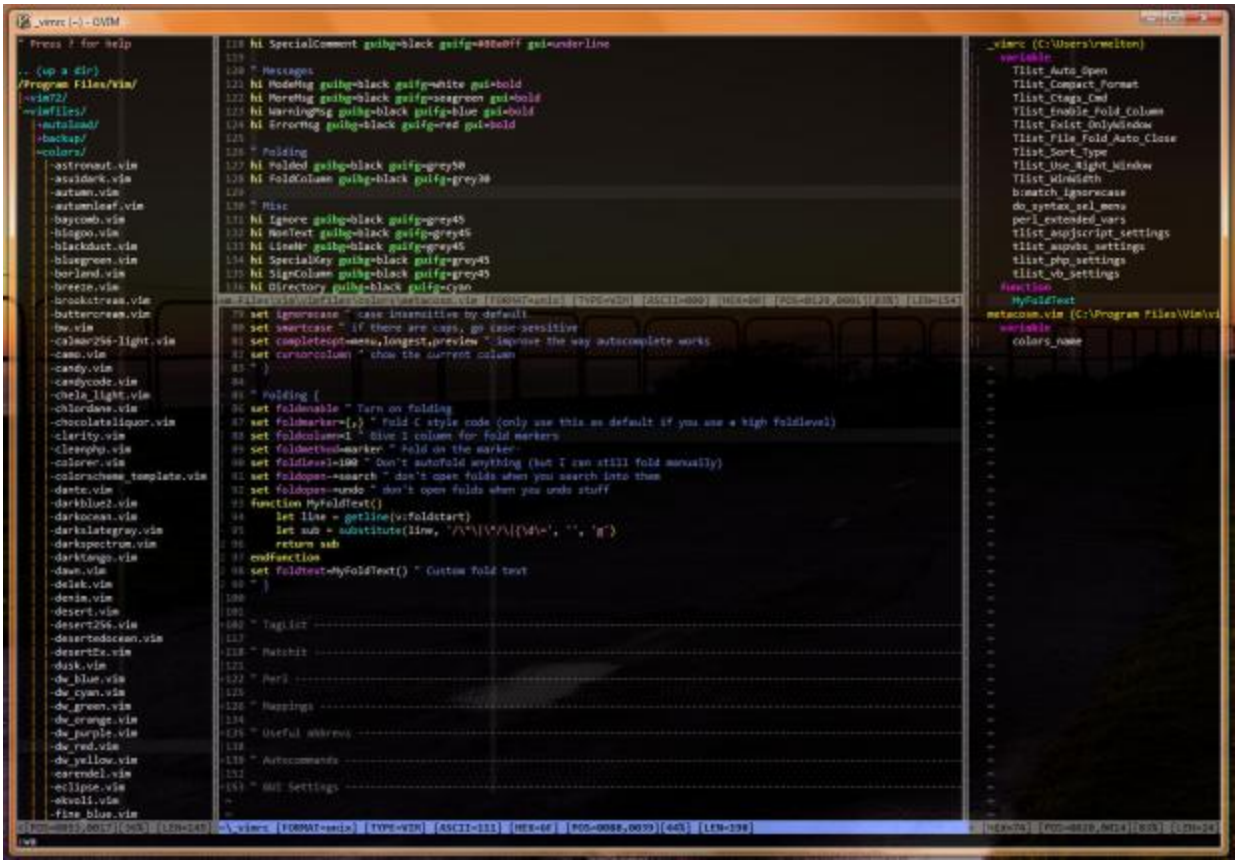
Cela fait beaucoup de questions dites donc.

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.



Opérations basiques (déplacement, écriture, enregistrement...)

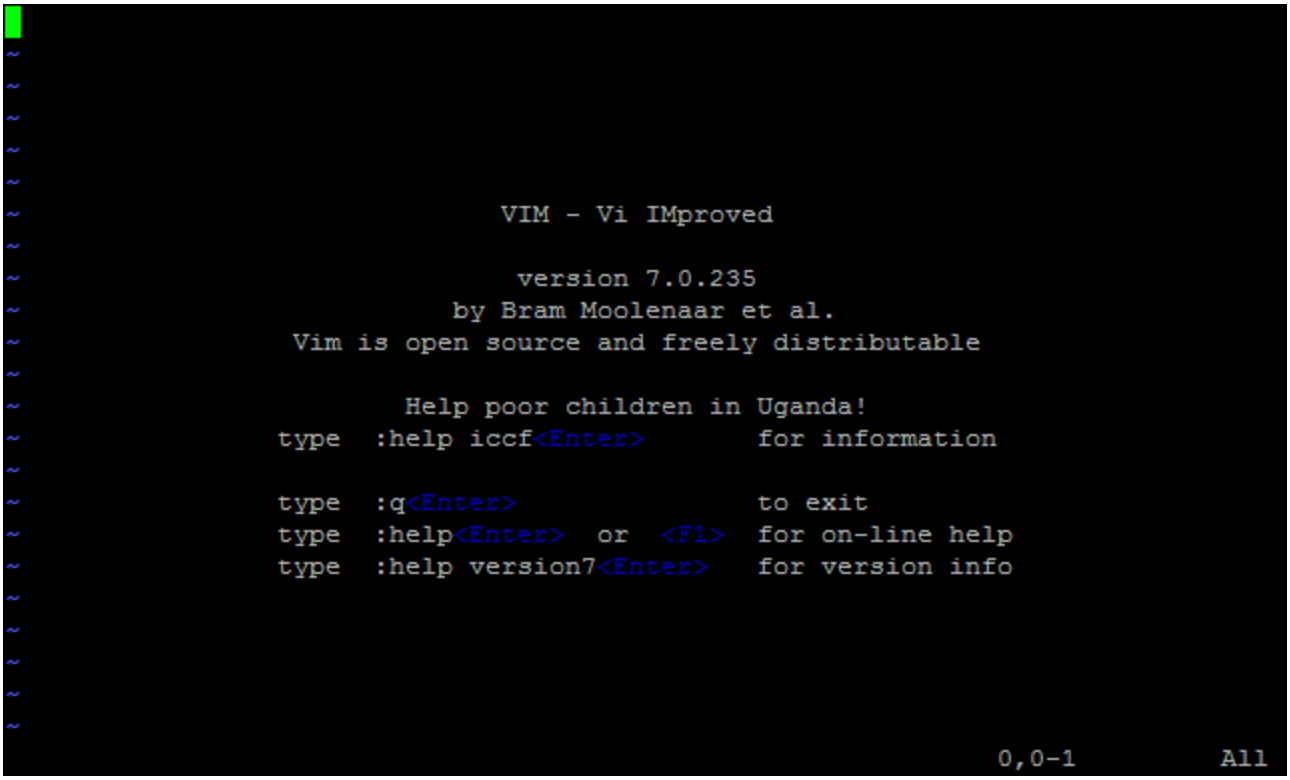
Nous allons découvrir Vim au travers de plusieurs étapes. Celles-ci deviendront de plus en plus complexes, mais nous allons commencer en douceur.

L'ouverture de Vim

Pour le moment, si vous lancez Vim en tapant simplement la commande suivante sans aucun paramètre :

```
vim
```

... il s'ouvre sur un nouveau fichier vide que vous avez déjà vu (figure suivante).



Vous pouvez aussi ouvrir un fichier en ajoutant son nom en paramètre :

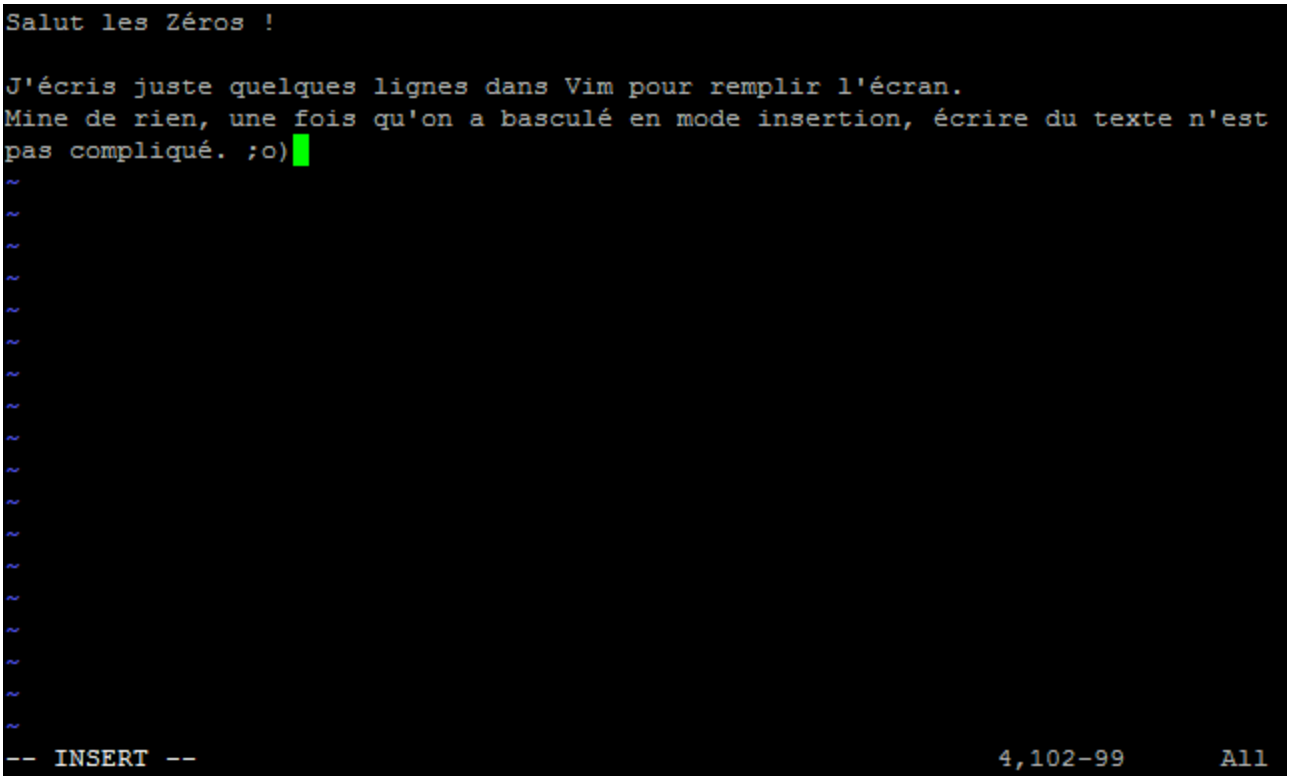
```
vim nomdufichier
```

Si le fichier n'existe pas, il sera créé.

`i` : insérer du texte

Nous allons partir d'un fichier vide. Nous souhaitons commencer par entrer du texte (quoi de plus normal pour un éditeur de texte, après tout ?).

Appuyez sur `i` (« i » minuscule). Vous basculez alors en mode insertion ; à présent, il vous est possible de taper du texte (figure suivante).



Notez le message `-- INSERT --` en bas de l'écran, qui vous confirme que vous êtes en mode insertion.

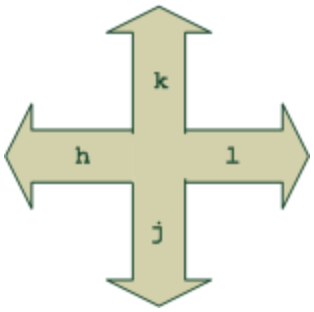
Écrivez quelques lignes comme moi puis appuyez sur la touche `Echap` pour revenir au mode interactif (le mode normal dans lequel vous vous trouviez au départ). Le message `-- INSERT --` disparaît alors et vous revoilà en mode interactif.

Le déplacement

`h` , `j` , `k` , `l` : se déplacer dans tous les sens

En mode interactif, il est possible de déplacer le curseur au sein du texte. Pour cela, on utilise les touches :

- `h` : aller à gauche ;
- `j` : aller en bas ;
- `k` : aller en haut ;
- `l` : aller à droite.



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, retenez 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

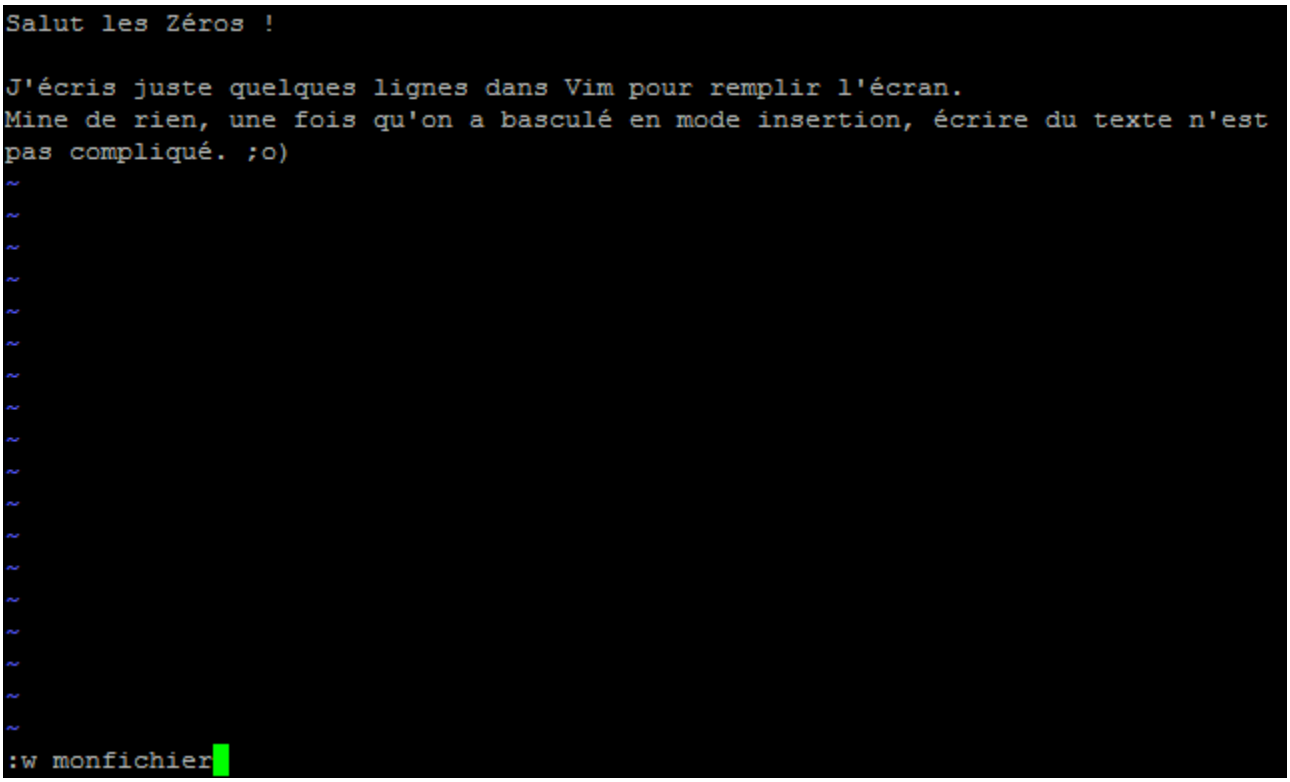
Pour enregistrer votre fichier, vous devez être au préalable en mode interactif (appuyez sur `Echap` pour vous en assurer).

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.



`:q` : quitter

Maintenant que vous avez enregistré, vous pouvez quitter Vim en tapant `:q` .

J'ai essayé de quitter en ayant fait des modifications après avoir enregistré et un message d'erreur s'affiche en rouge : « *No write since last change* ».

Vim vous interdit de quitter si vous n'avez pas enregistré vos changements. Vous pouvez toutefois forcer la fermeture du logiciel en ajoutant un point d'exclamation à la fin : `:q!` . Cette fois, il n'y aura aucune erreur.

`:wq` : enregistrer puis quitter

C'est la combinaison des deux commandes que nous venons de voir. Vous enregistrez et quittez immédiatement Vim lorsque vous tapez `:wq` .

Opérations standard (copier, coller, annuler...)

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.

`x` : effacer des lettres

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.

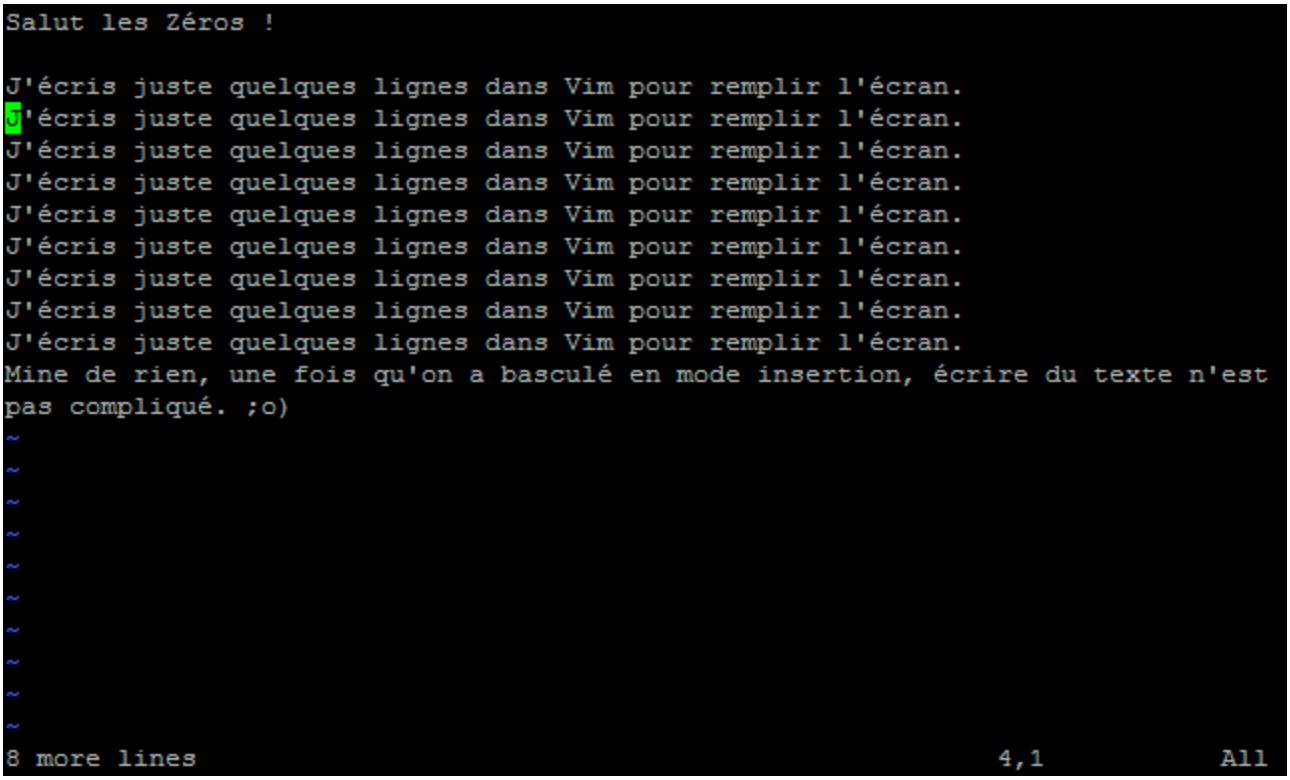
`p` : coller

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) !



`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` .

Opérations avancées (split, fusion, recherche...)

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.

/ : rechercher un mot

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`.

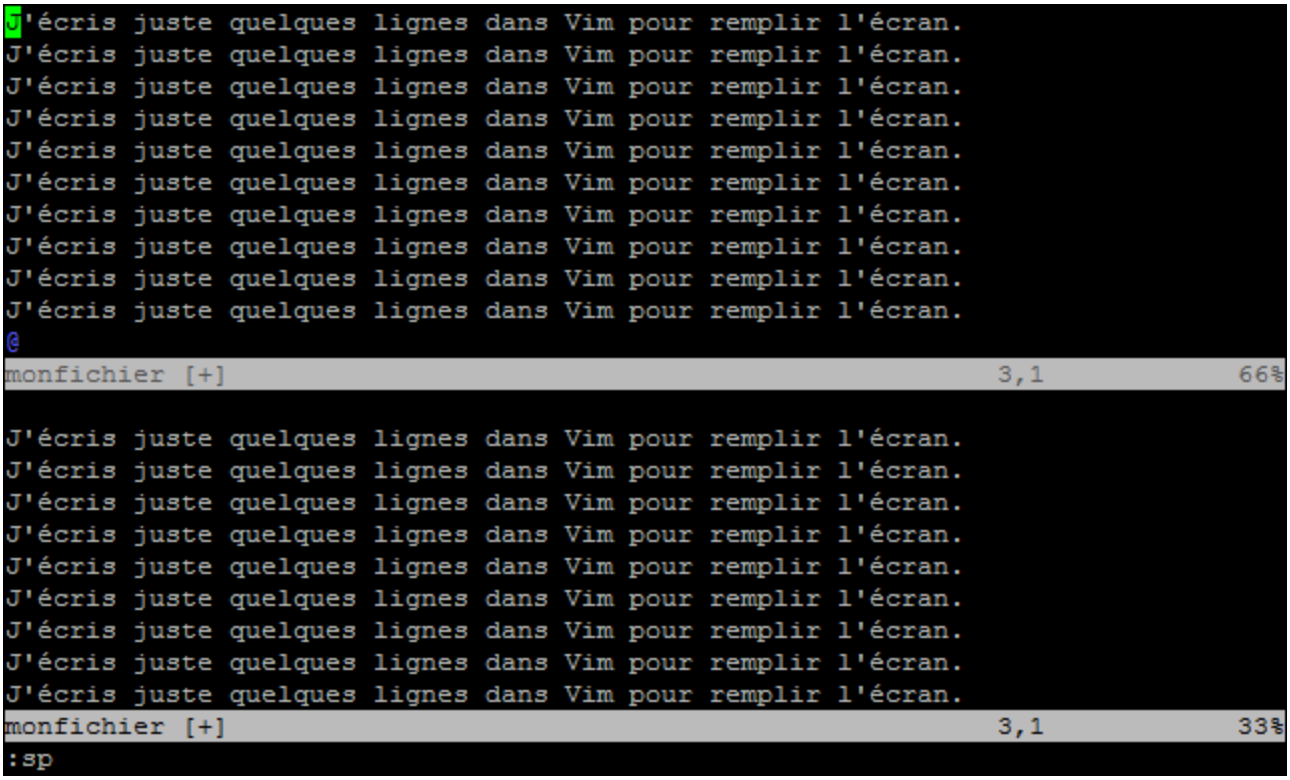
L'autocomplétion avec `Tab` fonctionne là aussi, donc pas besoin d'écrire le nom du fichier en entier !

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.

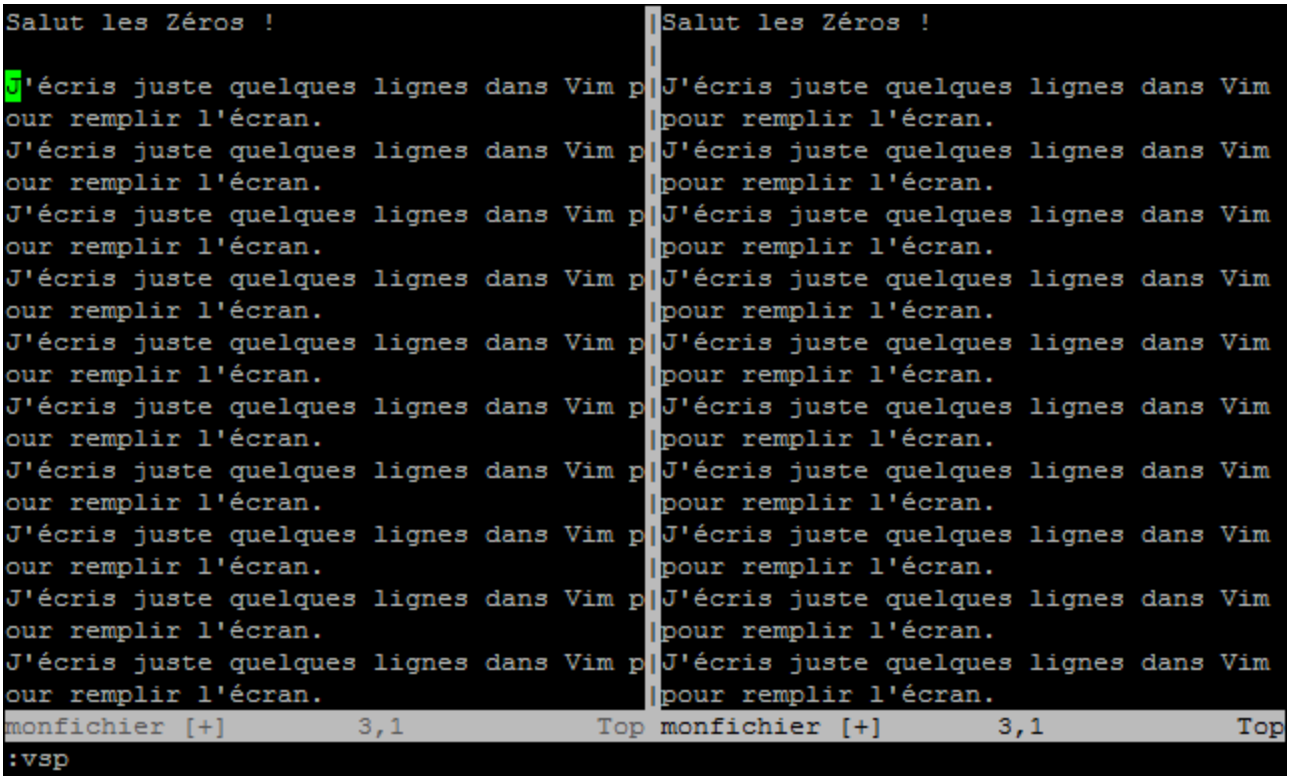


Le fichier est ouvert une seconde fois (ce qui vous permet de voir deux endroits différents du fichier à la fois) mais il est bien entendu possible d'ouvrir deux fichiers différents. Pour cela, ajoutez le nom du fichier à ouvrir à la suite de la commande : `:sp autrefichier` . Bonne nouvelle : l'autocomplétion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



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.

Les options de Vim

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.

Le fonctionnement des options

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
```

Notez qu'il faut enregistrer, quitter et relancer Vim pour que le changement soit pris en compte... sauf bien sûr si vous activez l'option à la volée en tapant dans Vim `:set syntax=ON`.

`background` : coloration sur un fond sombre

Par défaut, la coloration de Vim est plus adaptée aux fonds clairs. Les commentaires, par exemple, sont écrits en bleu foncé sur noir... ce qui n'est pas très lisible.

Si votre console est sur fond noir (comme chez moi), je vous recommande d'activer la prochaine option `background` et de la mettre à `dark`.

```
set background=dark
```

Les couleurs seront largement plus adaptées.

`number` : afficher les numéros de ligne

Il est possible d'afficher le numéro de chaque ligne à gauche (figure suivante) :

```
set number
```

Cela s'avère assez pratique, notamment quand on programme.

```
 1 Salut les Zéros !
 2
 3 J'écris juste quelques lignes dans Vim pour remplir l'écran.
 4 J'écris juste quelques lignes dans Vim pour remplir l'écran.
 5 J'écris juste quelques lignes dans Vim pour remplir l'écran.
 6 J'écris juste quelques lignes dans Vim pour remplir l'écran.
 7 J'écris juste quelques lignes dans Vim pour remplir l'écran.
 8 J'écris juste quelques lignes dans Vim pour remplir l'écran.
 9 J'écris juste quelques lignes dans Vim pour remplir l'écran.
10 J'écris juste quelques lignes dans Vim pour remplir l'écran.
11 J'écris juste quelques lignes dans Vim pour remplir l'écran.
12 J'écris juste quelques lignes dans Vim pour remplir l'écran.
13 Mine de rien, une fois qu'on a basculé en mode insertion, écrire du texte n'
   est pas compliqué. ;o)
~
~
~
~
~
~
~
~
~
                                                    6,1           Tout
```

Le

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

`nd` : afficher la commande en cours

Lorsque vous écrivez une commande comme `2dd` pour supprimer deux lignes, vous écrivez à [illegible] avez écrit.

Découvrez aussi ce cours en...

Co[illegible]Vi, Vim permet de pallier ce problème... encore faut-il activer l'option :


Livre


PDF

```
set showcmd
```

`ignorecase` : ignorer la casse lors de la recherche

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

OPENCLASSROOMS

```
set ignorecase
```

Qui sommes-nous ?

`mouse` : activer le support de la souris

Financements

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

Expérience de formation

Commencez par activer le support de cette dernière :

Forum

```
set mouse=a
```

Blog

Désormais, vous pourrez cliquer avec la souris sur une lettre pour y déplacer le curseur

Presse

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.

Que pensez-vous de ce cours ?

J'AI TERMINÉ CE CHAPITRE ET JE PASSE AU SUIVANT

QUIZ : QUIZ 4

INTRODUCTION AUX SCRIPTS SHELL