L'édition efficace avec Vim .

Cet article a été écrit à l'origine par Jonathan McPherson, la traduction est tirée du blog suivant :

http://fashion.hosmoz.net/blog/tag/vim

Je l'ai repris sous la beta2 de Word 2007 pour améliorer sa mise en page, notamment en présentant les raccourcis ou fonctions sous formes de tableaux. Il est évidemment possible qu'il reste ça et là des coquilles diverses, veuillez simplement me contacter et j'essayerai alors de rectifier cela le plus rapidement possible.

Kib².

Ce tutoriel requiert des connaissances de base de Vim -- mode insertion, mode commande, charger et sauver des fichiers, etc. Il a pour but d'aider les novices à développer leurs compétences pour utiliser Vim efficacement.

Dans ce tutoriel, <C-X> signifie Ctrl-X -- c'est à dire, gardez la touche Ctrl enfoncée et appuyez sur X. Vous pouvez obtenir de l'aide sur la plupart des commandes utilisées en tapant *: help command* dans Vim, en remplaçant command par ce sur quoi vous voulez de l'aide.

Se déplacer efficacement :

Rester en dehors du mode insertion :

En général, vous voulez passer le moins de temps possible dans le mode insertion de Vim, car ce mode fait agir Vim comme un stupide éditeur de texte. C'est pourquoi la plupart des novices passent tant de temps en mode insertion -- il rend Vim facile à utiliser. Mais la vraie puissance de Vim réside dans le mode commande ! Vous verrez que mieux vous connaitrez Vim, moins vous passerez de temps en mode insertion.

Utilisez h, j, k et l :

Le premier pas vers l'édition efficace avec Vim est de vous sevrer des touches *flèches*. Un des avantages du design de Vim est que vous n'avez pas constamment besoin de bouger vos mains entre les touches *flèches* et les touches *lettres*; quand vous êtes en mode commande, les lettres h, j, k et l correspondent aux directions gauche, bas, haut et gauche respectivement. Cela nécessite un peu d'entraînement pour s'y habituer, mais vous verrez la différence quand vous y serez habitué.

Quand vous éditez un e-mail ou un autre texte formaté par paragraphes, vous remarquerez peut-être que les touches de direction sautent plus de lignes que vous ne le voudriez. C'est parce que vos paragraphes apparaissent comme une seule longue ligne dans Vim. Tapez g avant h, j, k ou l pour bouger d'une ligne *écran* plutôt que d'une ligne *virtuelle*.

Utilisez les mouvements pour déplacer le curseur dans la ligne courante :

La plupart des éditeurs n'ont que des commandes simples pour déplacer le curseur (gauche, haut, droite, bas, au début/à la fin de la ligne, etc.). Vim possède des commandes de déplacement très avancée; ces commandes sont appelées mouvements (Ndr: *motions*). Quand le curseur bouge d'un point à un autre, le texte entre les points (points inclus) est considéré comme étant *parcouru* (C'est important pour plus tard).

Voici quelques uns des mouvements les plus utiles:

fx	Déplace le curseur en avant jusqu'à la prochaine occurrence du caractère x (bien entendu, x peut-être n'importe quel caractère). C'est une commande extrêmement utile. Vous pouvez taper ; pour répéter la dernière commande f.
tx	Pareil que f, mais bouge le curseur juste avant le caractère, pas juste dessus (Très utile, vraiment).
Fx	Déplace le curseur en arrière jusqu'à la prochaine occurrence de x sur la ligne courante.
w	Déplace le curseur en avant d'un mot.
b	Déplace le curseur en arrière d'un mot.
o	Déplace le curseur au début de la ligne courante.
^	Déplace le curseur sur le premier caractère de la ligne courante.
$	Déplace le curseur à la fin de la ligne courante.
(	Déplace le curseur en avant jusqu'à la prochaine phrase (Utile quand vous éditez un e-mail ou des documents textes).
)	Déplace le curseur en arrière jusqu'à la prochaine phrase.

Se déplacer efficacement dans le fichier édité :

Vim possède de nombreuses commandes qui peuvent vous transporter où vous le souhaitez dans votre fichier -- il est rare que l'on ait à le parcourir manuellement. Les combinaisons de touches ci-dessous ne correspondent pas techniquement à des déplacements, si on considère qu'elles servent à se déplacer dans le fichier et non pas dans une ligne.

<C-F>	Fait avancer le curseur à l'écran de texte suivant.
<C-B>	Fait reculer le curseur à l'écran de texte précédent.
numG	Place le curseur à la ligne num. (Par exemple, *10G* déplace à la ligne 10.)
gg	Déplace le curseur au début du fichier.
G	Déplace le curseur à la fin du fichier.
H	Déplace le curseur en haut de l'écran.
M	Déplace le curseur au milieu de l'écran.
L	Déplace le curseur en bas de l'écran.
*	Lis la chaîne de caractères sous le curseur et déplace ce dernier à l'endroit suivant où elle apparait de nouveau. (Par exemple, si votre curseur se trouvait sur le mot *bob*, le curseur se déplacerait jusqu'à la prochaine occurrence de *bob* dans votre fichier.)
#	Même chose que ci-dessus, sauf qu'il déplace le curseur sur l'occurrence précédente.
/text	A partir de la position du curseur, trouve l'occurence suivante de la chaîne de caractères *text* et s'y rend. Vous devrez presser la touche *Entrée* pour exécuter la recherche. Pour ré-exécuter votre dernière recherche, tapez *n* (pour *next occurance* (Ndr: occurrence suivante).)
?text	Même chose que /, mais cherche dans la direction opposée.

ma	Crée un signet nommé "a" référant à la position actuelle du curseur. Un signet peut être nommé par n'importe quelle lettre en minuscule. Vous ne voyez pas le signet, mais il est là !
`a	Aller au signet *a*. **Important**: c'est une backquote et non pas une quote simple. La touche backquote s'obtient à l'aide de la combinaison de touches Alt+7 (pavé principal) sur un clavier azerty.
` .	Aller à la dernière ligne que vous avez éditée. Cette fonction est très utile ! Si vous avez besoin de parcourir le fichier pour regarder quelque chose, vous pouvez revenir à l'endroit où vous vous trouviez sans avoir à utiliser de signet, en utilisant `.

Saisir efficacement :

Utiliser la complétion de mots :

Vim a un système de complétion de mots très pratique. Cela signifie que vous pouvez ne taper qu'une partie d'un long mot, presser une touche, et Vim fini de l'écrire pour vous. Par exemple, si vous avez une variable nommée *JeSuisUneVariableAvecUnNomImbuvable* quelque part dans votre code, vous ne voudrez probablement pas taper son nom en entier à chaque fois que vous souhaitez l'utiliser.

Pour utiliser la complétion de mots, tapez simplement les quelques premières lettres de la chaîne (pour notre exemple: JeSuisUn) et pressez <C-N> (cela signifie que vous devez maintenir enfoncé Ctrl et taper N) ou <C-P>. Si Vim ne vous donne pas le mot que vous souhaitez du premier coup, continuez d'essayer -- Vim vous proposera toutes les complétions possibles qu'il trouvera.

Entrez intelligemment dans le mode d'insertion :

La plupart des nouveaux utilisateurs de Vim, entrent dans le mode d'insertion en tapant *i*. Ca marche, mais c'est souvent inefficace car vi a un jeu de commandes permettant de basculer l'éditeur en mode d'insertion. Voici quelques unes des plus populaires:

i	Insère du texte à gauche du caractère courant.
I	Insère du texte au début de la ligne courante.
a	Insère du texte à droite du caractère courant.
A	Insère du texte à la fin de la ligne courante.
o	Crée une nouvelle ligne sous la ligne courante et y insère du texte.
O	Crée une nouvelle ligne au dessus de la ligne courante et y insère du texte.
c	{déplacement} Efface (change) le texte parcouru par le {déplacement} et insère du texte pour le remplacer. Par exemple, *c$* effacera le texte du curseur à la fin de la ligne et entrera en mode d'insertion. *ct!* effacera le texte à partir du curseur jusqu'au point d'exclamation suivant (sans l'inclure) et entrera en mode d'insertion. Le texte effacé est copié dans le presse-papier et peut être collé.
d	{motion} Efface le texte parcouru par le {déplacement} -- comme c{déplacement}, mais n'entre pas en mode d'insertion.

Déplacer efficacement des blocs de texte :

Utiliser la sélection visuelle et les modes de sélection appropriés :

A la différence de l'original vi, Vim vous laisse mettre en exergue du texte et y exécuter des opérations. Il y a trois modes de sélection visuelle principaux (ce sont les modes de mise en exergue). Ces modes sont les suivants:

v	Sélection *par caractère*. C'est le mode de sélection que la plupart des gens utilisent, donc entraînez-vous avec avant d'essayer les autres.
V	Sélection *par ligne*. Sélectionnera toujours des lignes entières. Plus pratique que le mode *par caractère* quand vous désirez copier ou déplacer un groupe de ligne.
<C-V>	Sélection *par bloc*. Extrêmement puissant et disponible dans très peu d'éditeurs. Vous pouvez sélectionner un bloc rectangulaire et tout le texte se trouvant à l'intérieur sera mis en exergue.

Toutes les touches de mouvement habituelles peuvent être utilisées -- donc, par exemple, *vwww* entrera dans le mode de sélection visuelle et mettra en exergue les trois mots suivants. *Vjj* entrera dans le mode de sélection visuelle *par ligne* et mettra en exergue la ligne courante et les deux sous cette dernière.

Couper et copier à partir de sélections visuelles :

Une fois que vous avez une sélection mise en exergue, vous voulez probablement en faire quelque chose. Quelques-unes des commandes les plus utiles que vous pouvez utiliser quand une partie d'un texte est mise en évidence:

d	Coupe (efface) le texte mis en exergue et le met dans le presse-papier.
y	Copie (ou *yank* (NdT: couper sec) qui est un terme Vim-èsque pour *copier*) le texte mis en exergue dans le presse-papier.
c	Coupe le texte mis en exergue et le met dans le presse-papier. Comme *d*, à l'exception qu'il laisse l'éditeur en mode d'insertion.

Couper et copier à partir de sélections non-visuelles :

Si vous savez exactement ce que vous voulez copier ou couper, vous pouvez le faire sans entrer dans le mode visuel. Vous gagnerez du temps.

d{déplacement}	Coupe le texte parcouru par le {déplacement} vers le presse-papier. Par exemple, *dw* coupera un mot et *dfS* coupera du curseur jusqu'à (en l'incluant) la prochaine lettre *S* en majuscule de la ligne courante.
y{déplacement}	Copie le texte parcouru par le {déplacement}.
c{déplacement}	Coupe le texte parcouru par le {déplacement} et laisse l'éditeur en mode d'insertion.
dd	Coupe la ligne courante.
yy	Copie la ligne courante.
cc	Coupe la ligne courante et laisse l'éditeur en mode d'insertion.
D	Coupe du curseur à la fin de la ligne courante.
Y	*Yank* la ligne entière, comme *yy*. (Oui, c'est contradictoire! Vous pouvez utiliser *y$* pour faire ce que vous attendiez d' *Y*.)
C	Coupe du curseur à la fin de la ligne courante et laisse l'éditeur en mode d'insertion.

x	Coupe le caractère courant. (C'est en quelque sorte la touche d'effacement arrière du *mode-commande*.)
s	Coupe le caractère courant et laisse l'éditeur en mode d'insertion.

Coller :

Coller est une chose simple. Mettez le curseur à l'endroit où vous voulez coller du texte et tapez *p*.

Utiliser plusieurs presse-papiers :

La plupart des éditeurs ont un seul presse-papier. Vim en a bien plus; les presse-papiers dans Vim sont appelés registres. Vous pouvez lister tous les registres définis actuellement et leurs contenus en tapant *: reg*. Typiquement, vous utiliserez des registres en lettres minuscules; les autres étant utilisés par Vim en internes et ne sont que rarement utiles.

Pour utiliser un registre spécifique pour une opération de type *copier/coller*, tapez simplement "a avant la commande, où a est le registre que vous voulez utiliser.

Par exemple, pour copier la ligne courante dans le registre k, vous pouvez taper "kyy. (Vous pouvez aussi taper V"ky. Pourquoi pas ?). Cette ligne restera dans le registre k jusqu'a ce que vous copiez spécifiquement quelque chose d'autre dans ce registre. Vous pouvez maintenant utiliser "kp pour coller le texte se trouvant dans le registre k.

Eviter les répétitions :

La stupéfiante commande :

Sous vi, taper . (un point) répétera la dernière commande que vous avez exécutée. Par exemple, si votre dernière commande était *dw* (effacement de mot (NdT: delete word)), vi effacera un autre mot.

Utiliser les compteurs :

Les compteurs sont une des plus puissantes et économiques (en temps) fonctionnalités de Vim. Nimporte quelle commande peut être précédée d'un nombre. Le nombre dira à Vim combien de fois exécuter la commande. Voici quelques exemples:

3j déplacera le curseur vers le bas, de trois lignes. 10dd effacera dix lignes.

y3&quote; copiera (*yank*) du curseur jusqu'à la troisième *quotation mark* (NDT: je laisse l'expression originale pour que vous puissiez comprendre la commande, mais il s'agit de guillemets) après le curseur, sur la ligne courante. Les compteurs sont utiles pour augmenter l'étendu d'un déplacement.

Enregistrement de macros :

Occasionnellement, vous vous retrouverez à faire la même chose encore et encore sur des blocs de texte de votre document. Vim vous laisse enregistrer une macro ad-hoc pour exécuter l'opération.

gregister	Commence l'enregistrement de macro dans le registre nommé register. Par exemple, *qa* commence l'enregistrement et met la macro dans le registre a. q : Fin de l'enregistrement.
@register	Rejoue la macro stockée dans le registre register. Par exemple, @a rejoue la macro du registre a.

Gardez en tête que les macros enregistrent simplement vos pressions de touches pour les rejouer ensuite. Il n'y a rien de magique. Enregistrer des macros est presque une forme d'art car il y a tellement de commandes qui accomplissent une tâche donnée sous Vim, et vous devez choisir avec attention les commandes que vous utilisez pendant que votre macro enregistre, afin qu'elles marchent aux endroits où vous avez prévu d'utiliser ladite macro.

Ecrire du code sous Vim :

Vim est un excellent éditeur de code source car il a de nombreuses fonctionnalités qui ont été spécifiquement conçues pour aider les développeurs. Voici quelques-unes des plus utiles:

]p	Comme pour *p*, mais ajuste automatiquement l'indentation du code collé pour correspondre au code dans lequel vous avez inséré. Essayez!
%	Mettre le curseur sur une accolade, un crochet ou une parenthèse et presser % enverront le curseur à l'accolade, au crochet ou à la parenthèse correspondant(e). C'est très pratique pour résoudre les problèmes de parsing liés à de multiples blocs de code imbriqués.
>>	Indente le code mis en exergue. (Voir la section plus haut, concernant la sélection efficace de texte. S'il n'y a pas de texte sélectionné, la ligne courante est indentée.)
<<	Comme >>, mais dés-indente.
gd	Aller à la ligne de définition (ou de déclaration) d'une variable.
k	Aller à la page de manuel pour le mot actuellement sous le curseur. (Par exemple, si votre curseur est sur le mot *sleep*, vous verrez la page de documentation de *sleep* s'afficher.)