L'éditeur Vi

Narmataru

December 6, 2006

Contents

1 Historique de Vi

Vi a été créé en 1976 par Bill Joy (c*o-fondateur de Sun Microsystems*) et signifie 'Visual'. C'est en fait la commande qui démarre le programme *ex* en mode plein écran (*ex étant un éditeur ligne à ligne basé sur ed*). Vi se prononce *Vi-aille* (*prononciation anglaise des deux lettre*). Il a été intégré à l'une des première version d'Unix BSD. Depuis, il, ou un de ses clones, est présent dans toutes les distributions Unix. La Single UNIX Specification (plus particulièrement l' ' IEEE standard 1003.2, Part 2: Shell and utilities ') inclut Vi comme éditeur de texte.

De nombreux clones de Vi ont vue le jour :

- nvi
- elvis
- **vim** (*La version distribué dans la majorité des distributions Linux*)
- Vile (tente de réconciliers les utilisateurs de Vi et d'Emacs en gardant le meilleur des deux éditeurs)
- *etc...*

2 Les différents modes

Une des particularités de Vi est que c'est un éditeur modal. Celà signifie que les touches du clavier qui permettent de faire l'édition, changent de signification selon le mode dans lequel il se trouve. Ainsi Vi possède 4 modes d'utilisation :

- Le mode commande ou normal, qui est celui dans lequel il se trouve lors de son ouverture
- Le mode insertion qui permet d'éditer du texte un à la manière des éditeurs classiques
- Le mode remplacement qui *écrase* le texte existant
- Le mode ex qui est en faite l'utilisation de Vi en mode ligne (*aussi appelé command-line*)

La touche 'ECHAP' permet toujours de revenir au mode commande. C'est à partir de celui-ci que nous pouvons choisir un des autres modes. La touche 'i' permet d'éntrer dans le mode édition et la touche 'R' dans le mode remplacement (*d'autres touches permette d'entrer dans ces modes de manière un peu différentes*), et la touche ':' rentre dans le mode ex.

2.1 Le mode insertion

Ce mode est certainement celui que recherchera un néophite. En effet, il permet l'insertion et la suppression de texte un peu à la manière du fameux bloc-note de MS-Windows©. Pour aller dans ce mode mettez-vous en mode commande (*si vous n'êtes par sur d'y être appuyez plusieurs fois sur ECHAP)* puis appuyez sur :

- *i* (*insertion*) pour rentrer en mode insertion juste avant le curseur
- *I* pour rentrer en mode insertion en début de ligne

- *a* (*after*) pour rentrer en mode édition après le curseur
- *A* pour rentrer en mode insertion à la fin de la ligne
- *o* Insère une ligne sous le curseur et rentre en mode édition
- *O* Insère une ligne au dessus du curseur et rentre en mode édition

Pour revenir au mode commande apuyez simplement sur *ECHAP*.

2.1.1 La complétion automatique

Vim permet de compléter automatiquement l'écriture de texte. Vim recherche les candidats à la complétion dans tous les mots de tous les fichiers ouvert. Ainsi, en mode insertion, vous pouvez taper les première lettre d'un mot puis utiliser *CTRL+p* ou *CTRL+n* pour faire une recherche de complétion vers l'avant ou vers l'arrière. Appuyez plusieurs fois sur le racourcis pour trouver le bon mot.

A partir de la version 7 de Vim, les mots candidats à la complétion sont affichés dans une boite de texte et on peut choisir le mot avec les flèches directionnelles.

Vim peut donc compléter automatiquement un mot. Mais comme à son habitude il en peut encore plus !

- *CTRL+x CTRL+l* : complète une ligne entière
- *CTRL+x CTRL+k* : complète avec un mot du dictionnaire sélectionné. *:set dict=nomDuDictionnaire* permet de sélectionner un dictionnaire

2.2 Le mode remplacement

Le mode remplacement permet d'écrire *par dessus* du texte existant. L'intérêt réside dans le fait que la touche BACKSPACE ne supprime pas bêtement les caratères mais remet les caratères initiaux.

On rentre dans ce mode à partir du mode commande par la touche 'R' et on revient au mode commande via *ECHAP*.

2.3 Le mode commande

Il s'agit du mode de fonctionnement de Vi par défaut. C'est surement celui qui rebute le plus les débutant.

2.3.1 Déplacement du curseur

Vi a été développé pour être utilisé via un terminal relié à un serveur Unix. Or ces terminaux n'avait pas forcément de touches directionnelles. Donc les touches originelles pour déplacer le curseur sont :

- *h* pour aller à gauche
- *j* pour descendre d'une ligne
- *k* pour monter d'une ligne
- *l* pour aller vers la droite

Certains clone de Vi peuvent cependant utiliser les touches directionnelles pour dirirger le curseur. C'est le cas de Vim.

Contrairement à la majorité des éditeurs, Vi ne se content pas de déplacer le curseur caractère par caractère. Il possède notamment la notion de ligne, de mot, de phrase, de paragraphe et de section.

Vous pouvez toujorus préfixer une commande déplacement avec un chiffre. Ainsi, vous vous déplacerez plus vite. Par exemple *5h* déplacera le curseur de 5 caractères vers la gauche.

Déplacements par ligne

- ^ déplace le curseur en début de ligne
- *$* déplace le curseur en fin de ligne
- *+* premier caractère non blanc de la ligne suivante
- - premier caractère non blanc de la ligne précédente
- *H* première ligne de l'écran
- *M* ligne médiane de l'écran
- *L* dernière ligne de l'écran
- *n|* déplace le curseur à la colonne *n* de la ligne courante

Déplacements par mot

- *w* (*word)* déplace le curseur au premier caractère du mot suivant
- *b* (*before*) déplace le curseur au premier caractère du mot précédent
- *e* (*end*) déplace le curseur à la fin du mot suivant

Autres déplacements

- *)* *ou* *(* dépace le curseur à la phrase suivante, précédente
- } *ou* { déplace le curseur au paragraphe suivant précédent
-]] *ou* [[déplace le curseur à la section suivante, précédente
- *:n<ENTREE>* va à la ligne *n* du fichier
- *gg* va la première ligne du fichier
- *SHIFT+g* va à la dernière ligne du fichier
- *fn* va à la prochaine occurance du caractère *n* dans la ligne (*ex : f(va sur la prochaine parenthèse*)
- *Fn* va l'occurence précédente du caractère *n* sur la ligne.

Les marqueurs

Un marqueur permet de sauvegarder l'emplacement du curseur pour y revenir. Un marqueur est défini par une lettre. Vous définissez un marqueur via la commande *mn* ou *n* est l'identifiant. Pour y revenir faites *'n*

Voici un exemple :

- *vim /etc/fstab* j'ouvre le fichier fstabe
- *:5* je vais à la ligne 5
- *ma* je défini le marqueur *a*
- *:7* je vais à la ligne 7
- *mz* je place le marqueur *z*
- *gg* je vais à la première ligne
- *'z* je retourne au marqueur *z* en ligne 7
- *'a* je retourne au marqueur *a* en ligne 5

2.3.2 Commandes d'édition

Vi et plus particulièrement Vim possède énormément de commande facilitant l'édition. En voici un bref aperçue.

Commandes de suppression

La commande de suppression est '*d*'. Il faut associer cette commande avec une autre pour spécifier ce que l'on veut supprimer :

- *dd* supprime la ligne courante
- *dw* supprime le mot suivant
- *db* supprime le mot précédent
- *dh* supprime le caractère de gauche
- *d$* supprime de la position courante jusqu'à la fin de la ligne
- *d^* supprime de la position courante jusqu'au début de la ligne
- *dfp* supprime jusqu'à la prochaine occurence de *p* sur la ligne
- *etc... vous associez d avec une commande de déplacement*

Vous pouvez aussi définir le nombre de suppression a effectuer en mettant un chiffre devant. Par exemple '*5dw*' supprime les 5 prochains mot.

'*x*' est un raccourcis pour '*dl*' qui supprime le caractère sous le curseur.

Les commandes de suppression ne font pas que supprimer. En faite elles coupent comme nous le ferions avec un classique CTRL-X et place le texte dans le tampon par défaut.

- *p* ou *P* permettent de recoller le texte supprimer (*pour plus de détailles allez vous la section 'Le copier/coller'*)

Ainsi '*xp*' interverti 2 caractères.

Le copier/coller

La commande pour faire une copie de texte est '*y*' et '*p*' pour coller. Comme pour la suppression, vous pouvez associer une commande de déplacement.

- *yy* copie la ligne courante
- *yw* copie le mot suivant
- *yb* copie le mot précédent
- *etc...*

'p' permet de copier après tandis que '*P*' copie avant la position.

Un autre des grand atouts de Vi est ses tampons de sauvegarde. Nous sommes tous habitués au CTRL+C/CTRL-V or ce système ne permet de ne faire qu'une copie à la fois dans le seul et unique tampon de sauvegarde système. Or sous Vi il y a une multitude de tampons de sauvegarde. Ainsi, on peut copier plusieurs zones de texte en même temps et les réutiliser comme bon nous semble. Pour sélectionner un tampon la commande est '*'t*' ou *t* est une lettre désignant le tamon que l'on veut utiliser. Un tampon peut être désigné par une des lettres minuscules de l'alaphabet [a-z]. Ainsi on a :

- *'ayy* copie la ligne courante dans le tampon *a*
- *'byw* copie le mot suivant dans le tampon *b*
- *etc...*

Pour coller un tampon spécifique il suffit de sélectionner le tampon et d'appeler la commande '*p*' ou '*P*' :

- *'ap* colle le tampon *a* après le curseur
- *'bP* colle le tampon *b* avant le curseur

Mais pourquoi seule les lettres minuscules peuvent désigner un tampon et non leurs grandes soeurs les majuscule ? Ces dernières servent à ajouter du texte dans le tampons !

- *'Ayy* ajoute la ligne courante dans le tampon *a*
- *'Zyw* ajoute le mot suivant dans le tampon *z*

Il y a 2 tampons un peut spéciaux. Le premier est le tampon par défaut. C'est celui qui est sélectionné si on utilise pas la commande *'n*. Le second tampon un peu spécial est le tampon désigné par '+'. En effet, il s'agit du tampon système. grace à lui Vi peut coller du texte sélectionné à partir d'une autre application ou en copier.

La sélection de texte

Vim permet de sélectionner toute une zone de texte sur laquel l'éditeur focalisera ses commandes. Dans le jargon de Vi celà s'appelle le mode visuel.

- *<SHIFT>+v* : permet un sélection ligne à ligne

- <CTRL>+v : sélectionne en bloque

Lorsque le mode de sélection est activé, utilisé les commandes de déplacement.

Une fois le texte sélectionné, les commande que vous effectuerez travailleront sur cette sélection.

La sélection en mode bloc est particulièrement pratique car elle permet de traiter une partie de plusieurs ligne en même temps.

Figure 2.1: Modification de plusieurs ligne en même temps

Commandes de modification

La commande *c* permet d'effacer une entité et de rentrer dans le mode édition :

- *cw* efface le mot suivant et rentre en mode édition
- *cc* efface la ligne et rentre en mode édition
- *etc...*

Pour changer la casse de caractères il faut utiliser la commande ~. Cette commande utilisée seule change la casse du caractère sous le curseur. On peut rentrer en mode sélection pour modifier la casse de plusieurs mots ou lignes :

- *<CTRL>+v e ~* sélectionne un mot et change sa casse

Undo et redo

Comme tout bon éditeur, Vi possède une commande pour annuler et un autre pour refaire les modifications

- *u* undo, annule les dernières modifications du fichier
- *CTRL-SHIFT-r* redo, refait les modifications

2.3.3 Recherche de texte

De la même manière qu'il existe plusieurs types de déplacement, il y a plusieurs manière de faire une recherche. Un des grands atouts de vi c'est que l'on peut utiliser les expression rationnelle pour les recherches. Les expressions rationnelles de Vim diffèrent un peu de celles de perl dans le fait qu'il faut ajoute plus d'anti-slash '\'.

La commande la plus utile pour faire une recherche est '/'. Lorsque vous appuyez sur cette touche en mode commande, un slash '/' apparait en bas à gauche de l'éditeur. Vous pouvez désormais saisir une expression rationnelle et la valider en appuyant sur ENTREE.

- */bonjour* : recherche le mot *bonjour*
- */[bB]onjour* : recherche le mot *bonjour* ou *Bonjour*
- */[bB]onjour\|monde* : recherche un de ces mots : *bonjour, Bonjour ou monde*

Dans Vim il est bon d'activer les option *hlsearch* et *incsearch* pour améliorer l'affichage des recherches.

La commande '*?*' fonctionne de la même manière sauf que la recherche se fait vers l'arrière. Enfin la touche '*n*' permet d'aller à la concordance suivante tandis que '*N*' va la précédente (*en respectant le sens de la recherche*).

- ** ou #* recherche, vers l'avant ou l'arrière, la prochaine occurence du mot sur lequel se trouve le curseur

- *fx ou Fx* recherche vers l'avant ou l'arrière, le caractère *x* dans la ligne courante. *';*' ou ',' répète la recherche.

Dans cette capture vous pouvez voir le résultat de la dernière recherche surligné en marron, et la construction de la nouvelle recherche en vert. Les options hlsearch (surlignage du resultat) et incsearch (affichage instantané du résultat de la recherche) sont activées. ':set hlsearch' ':set incsearch'

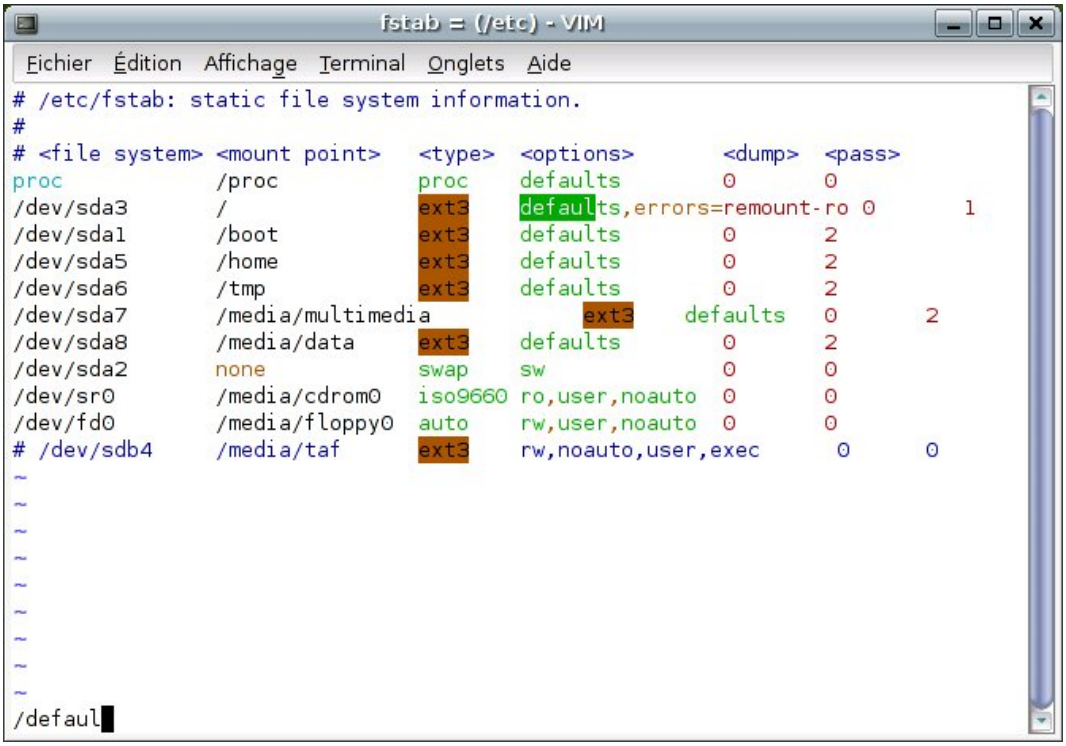

Figure 2.2: Vim en pleine recherche

2.3.4 Créer une macro

Vim peut enregistrer les différentes action que vous faites et les refaire automatiquement à la demande. Ceci permet par exemple de traiter un gros fichier en ne modifiant réellement qu'une seule ligne. On enregistre les commandes de modification d'un ligne (la première par exemple), puis on dit d'exécuter la macro sur les autres lignes.

- *qa* enregistre les actions dans le tampon *a (Il faut appuyer sur 'q' pour arreter l'enregistrement)*

- *@a* exécutes la macro *a*

Vous pouvez aussi créer une macro de toute pièce que vous affecterez à une touche :

- *:noremap <C-h> ^:s/public/private/<ESC>* cette exemple va affecter à la combinaison de touche *CTRL+h* la commande ^ qui va en début de ligne, puis la commande de substitution qui remplace *public* par *private*.
- *:noremap <C-k> opublic void main(String[] args){<CR>}<ESC>* Cette commande associe à *CTRL+k* la création de la méthode main d'un programme java. La première commande *o* permet de créer une ligne vide en dessous du curseur et rentre en mode insertion.

Il est aussi possible à Vim de modifier à la volée le texte que vous saisissez :

- *:inoremap narma narmataru* Dès que vous écrirez *narma* Vim remplacera par *narmataru*
- *:inoremap test <ESC>:s/main/main_old/<CR>A// ancienne méthode main* cette exemple permet dès que l'on saisi le texte *test* de modifier *main* en *main_old* et de rajouter un commentaire en fin de ligne. Le mot *test* doit être saisie n'importe ou sur la ligne contenant le mot *main*.

Pour voir toutes les commande de *mappage* tappez *:h :noremap*

2.3.5 La composition de commandes

Souvent les commandes peuvent être combinée entre elles pour faire exactement ce que l'on souhaite :

- *dfx* supprime tous les caractères jusqu'au prochain *x*
- *cf(* modifie le texte jusqu'à la prochaine parenthèse
- *'ayf(* copie le texte jusqu'à la prochaine parenthèse dans le tampon *a*
- *'z2w* copie les 2 prochains mots dans le tampon z

2.4 Le mode ex

Le mode ex est appelé par la commande ':'. Vous pouvez aussi directement appeler la commande ex depuis un shell '*ex /etc/fstab*' (*q<ENTREE> pour quitter*) par exemple. Il s'agit d'un éditeur en ligne à ligne. Ceci était bien pratique il y a quelques années lorsque les terminaux était encore relié au serveurs Unix, que les réseaux étaient lent et que la RAM coutait très chère. Ainsi, si ou ouvrait un fichier avec ex, peut de données transitaient sur les réseaux (*la ou les lignes d'édition et les commandes*).

Donc Vi est le mode visuel de ex. Lorsque vi est lancé on peut utiliser les commande ex via la commande ':'. Lorsque vous rentrez cette commande les deux points s'affiche en bas à gauche de l'éditeur. Vous pouvez alors saisir les commandes ex désirées. Ces commandes sont exécutées lorsque ENTREE est appuyé.

Une des commandes les plus importante permet de quitter l'éditeur '*:q<ENTREE>*'

Dans ce mode le caractère '%' sera remplacé par le nom du fichier ouvert. Par exemple '*:w %.bak*' fera une sauvegarde du fichier ouvert avec l'extension *.bak*.

2.4.1 Ouvrir et sauvegarder des fichiers

Vi peut ouvrir plusieurs fichiers en même temps. Il dispose de plusieurs tampons. Pour ouvrir un seule fichier faite '*vi /etc/fstab*' et pour en ouvrir plusieurs '*vi /etc/fstab /etc/X11/xorg.conf*'. Un fois que Vi est lancé, vous pouvez voir les tampons existant via la commande ':ls'. Vous pouvez ouvrir un nouveau fichier avec '*:e /le/fichier/a/ouvir.txt*'. La commande '*:ls*' permet de lister les différents tampons et '*:bn*' permet de sélectionner le tampon n°*n*. Pour sauvegarder un tampon (*fichier*) utilisez la commande '*:w* [*nomDuFichier*]'.

- :*e nomDeFichier* ouvre un fichier dans un nouveau tampon
- *:ls* liste les tampons d'édition
- *bn* affiche le tampon *n* et permet son édition
- *:w*[*!*] [*nomDefichier*] sauvegarde le tampon d'édition dans le fichier d'origine ou dans *nomDeFichier*. *!* outrepasse le *readonly* (*force l'écriture*).
- *:q*[*!*] quitte l'éditeur. *!* force la fermeture si le fichier a été modifié

Ouvrir plusieurs fichiers en même temps

La fenêtre de Vim peut ête divisée en plusieurs partie chacune d'elle permettant d'afficher un tampon fichier :

- *:new nomDuFichier* divise horizontalement la section en cours et ouvre *nomDuFichier*
- *:vnew nomDuFichier* divise verticalement la section en cours et ouvre *nomDuFichier*
- *CTRL+W+W* : active le curseur dans la prochaine division
- *CTRL+W FLECHE DIR.* : active le curseur dans la division de la direction donnée
- [*n*] *CTRL+W++* : agrandi la division de *n* lignes (ou 1 si *n* n'est pas spécifié)
- [*n*] *CTRL+W+-* : diminue la division de *n* lignes (ou 1 si *n* n'est pas spécifié)
- *CTRL+W+_* : agrandi la division au maximum
- *:q* : ferme le fichier et la division active

L'éditeur graphique *gvim* permet même de gérer les onglet (*à partir de la version 7*):

- *:tabnew nomDuFichier* : ouvre *nomDufichier* dans un nouvel onglet

2.4.2 Substitution de texte

La commande de substitution du mode ex est '*s*'. il faut spécifier l'expression rationnelle et la chaine de remplacement. D'autres options permettent de personnaliser la substituion :

- *s/motif/remplace/* remplace la première occurence de *motif* sur la ligne, par *remplace*
- *s/motif/remplace/g* remplace toutes (*g*) les occurences de *motif* sur la ligne par *remplace*
- *%s/motif/remplace/g* remplace toutes (*g*) les occurences de *motif* dans tout le fichier (*%*) par *remplace*

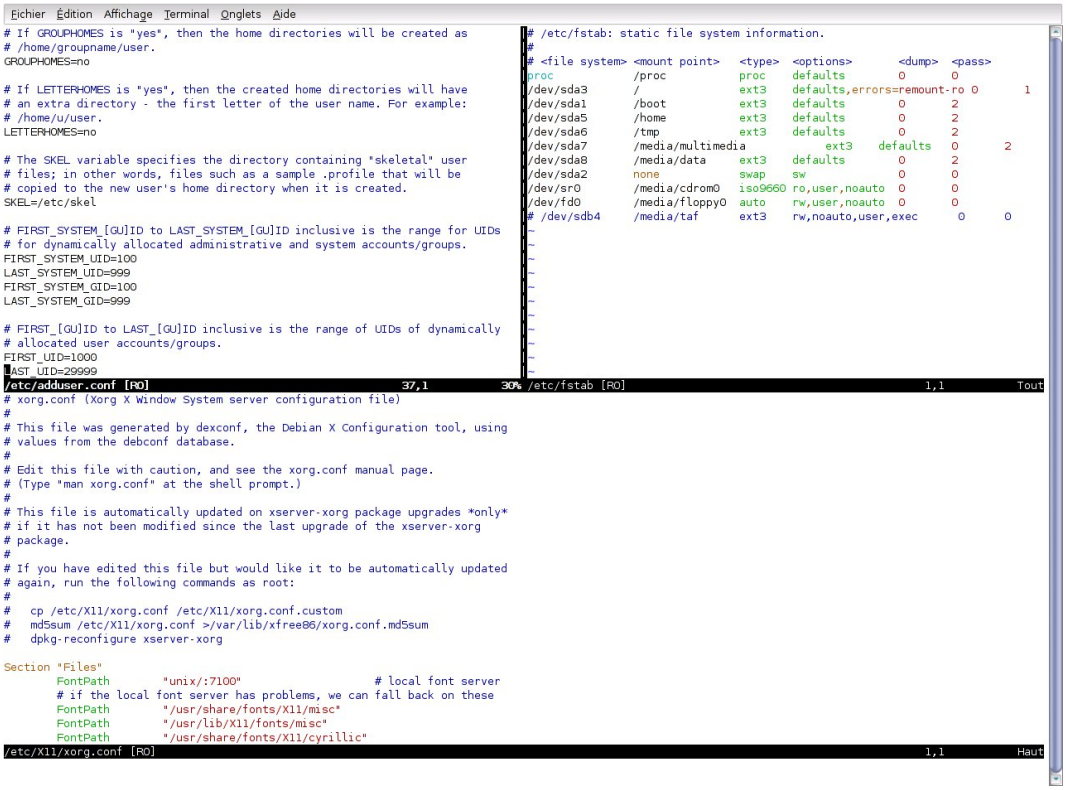

Figure 2.3: Vim permet d'afficher plusieurs fichiers en même temps

- *%s/motif/remplace/gc* remplace toutes (*g)* les occurences de *motif* dans tout le fichier (*%*) par *remplace* et demande confirmation (*c*)
- *n,ms/motif/remplace/gc* remplace toutes (*g*) les occurences de *motif* de la ligne *n* à la ligne *m* avec confirmation (*g*)

Evidemment ce ne sont que des exemples, toutes ces options peuvent être mélangées et il en existe d'autres. Si vous avez sélection du texte avec *CTRL+V* ou *SHIFT-V*, la commande de substitution ne travaillera que sur les blocs sélectionnées.

2.4.3 Paramétrage de l'éditeur

L'éditeur Vi est très configurable. Il existe énormément d'options paramétrables.

Les flags

Les flags sont des options qui ne peuvent prendre que 2 valeurs :

- activé
- désactivé

Pour activer un flag on utilise la commande '*:set nomduflag*' et pour le désactiver '*:set* ***no****nomduflag*'.

- *hlsearch* : surligne le résultat d'une recherche
- *incsearch* : affiche le résultat de la recherche pendant son élaboration

Activation/désactivation de paramètres

par exemple :

- *:syntax* [*on*|*off*] : active ou désactive la coloration syntaxique

2.4.4 Exécuter des commandes externes

Vi permet d'exécuter des commandes externes et même d'intégrer leur sortie dans le fichier en cours d'édition :

- *:r!ls* insère la sortie du *ls* dans le fichier ouvert

- *:!javac @jc_argfile %* compile le fichier ouvert via le compilateur java avec les options contenues dans *jc_argfile*
- *:!!* reexécute la dernière commande
- Vous pouvez remarquer que *%* remplace le nom du fichier ouvert.

3 Utilisation de l'aide intégrée

Vim possède une aide intégré très riche. Elle s'obtient via la commande ex '*:h*[*help*]'. Vous pouvez spécifier sur quel point vous souhaitez de l'aide :

- *:h :h* : donne l'aide sur l'aide
- *:h :s* : donne l'aide sur la commande de substitution
- *:h /* : aide sur la recherche de motif
- *:h d* : l'aide sur la suppression

Comme vous pouvez le constater, il suffit de faire suivre *:h* de la commande sur laquel on a besoin d'aide.

4 La personnalisation de l'éditeur via le fichier .vimrc

Vim possède plusieurs fichiers qui seront lus/interprétés lors de son démarrage. Ces fichiers permettent d'activer certaines option, de créer des racourcis clavier, des macros, etc...

Le principal fichier qui permet la configuration de Vim est *.vimrc*. Il en existe deux ! Un qui configure Vim pour tout le monde. Les Vim de tous les utilisateur le liront. Le second est propre a chacun et c'est celui qu'il est conseillé de modifier.

Sous linux (*et je pense les autre unices*) le fichier de configuration général est */etc/vim/vimrc*. La configuration personnelle se trouve dans ~/.vimrc (~ *est la racine du compte utilisateur*). Ces fichiers existent aussi sous windows et fonctionne de la même manière.

5 Liens externes

Tout sur Vim, téléchargement de script http://vim.org
Vi sur wikipedia http://fr.wikipedia.org/wiki/Vi_%28logiciel%29
L'éditeur ed, la source de Vi et ex http://fr.wikipedia.org/wiki/Ed_%28logiciel%29
Le guide de survie de l'utilisateur de Vi http://matrix.samizdat.net/pratique/documentation/guide-survie-VI.html
Introduction à Vim http://www.linux-france.org/article/appli/vi/vim/
How-to de l'éditeur Vim http://fr.tldp.org/HOWTO/telechargement/html-1page/Vim-HOWTO.html.gz
Une présentation de notre ami Luc Hermitte http://hermitte.free.fr/vim/ressources/vim-config.pdf

6 entete

<entete>
<meta>
<description>Ceci est un article sur Vi et plus particulièrement Vim</description>
<keywords>Vi,Vim,editeur,texte</keywords>
</meta>
</entete>