VIm: Vi Improved

Anzala-Yamajako Alitcha

June 4, 2008

Table des matières

1 Pourquoi VIm?

Avant de commencer il convient quand même de se demander á qui il est utile d'apprendre á se servir de VIm. D'après moi, toute personne qui estime ne pas utiliser toutes les capacités de son éditeur de texte actuel (que ce soit notepad ou emacs) à beaucoup à gagner à apprendre VIm. Il existe une foultitude de commande VIm et on ne se sert pas de toutes tous les jours mais lorsque qu'on le fait l'efficaté est souvent impressionante. L'ambition de ce tutoriel est surtout de donner les bases et ce qui me semble utile. En passant il me semble qu'il n'est pas idiot de rappeler que VIm existe aussi bien sous linux que sous Windows ou Mac OS.

2 Conventions

Voici les conventions d'écriture de ce document (ce sont celles en vigueur sur lea-linux et je les trouve particulièrement claires).

- Les touches à taper sont en gras et si elles n'apparaissent pas dans la lignes de commandes
- *n* est un entier qu'on remplacera par n=2,3... il est omis si *n*=1. La plupart des commandes de VIm peuvent être précédées de cet entier qui signifie qu'on l'executera *n* fois (très utile). Exemple: **[2dd]** supprime la ligne courante et celle d'après.
- *x,y* reprsente une plage de ligne. Notez que on peut omettre *x* ou *y* l'élément omis valant alors le numero de la ligne courante. On peut aussi remplacer *y* par *$* qui désigne alors la fin du fichier et finalement *x,y* peut être remplacé par % qui signifie *1,$* donc l'ensemble du fichier.

  Exemple:

  - **:10,20s/p=p+1/p++/** effectue la substitution de p=p+1 par p++ entre les lignes 10 et 20.
  - **:,20s/p=p+1/p++/** effectue la substitution de p=p+1 par p++ entre la ligne courante et la ligne 20.
  - **:10,s/p=p+1/p++/** effectue la substitution de p=p+1 par p++ entre la ligne 10 et la ligne courante.
  - **:10,$s/p=p+1/p++/** effectue la substitution de p=p+1 par p++ entre la ligne 10 et la fin du fichier.
  - **:%s/p=p+1/p++/** effectue la substitution de p=p+1 par p++ sur tout le fichier.

3 Editeur multi-modal

Il convient de rappeler que VIm est editeur de texte à 2 modes, le mode commande dans lequel on rentre en pressant [**Echap**] et le mode insertion dans lequel on rentre avec

- [**i**],[**I**] dans le premier cas la saisie se fait alors sur la position du curseur dans le second au début.
- [**a**],[**A**] dans le premier cas la saisie se fait après le curseur dans le second à la fin de la ligne.
- [**o**],[**O**] dans le premier cas la saisie se fait à la ligne suivant celle sur laquelle se trouve le curseur dans le second elle se fait avant.
- [**R**] Comme [**i**] sauf que la saisie se fait en remplacement (équivalent de la touche insert).
- [**r**]***lettre*** Remplace le caractère sous le curseur par *lettre*.
- [**c**] remplace l'ancien texte avec le nouveau. Elle est suivie d'argument de déplacement. On verra des exemples dans les parties suivantes mais retenez qu'elle est très utile.

4 Les choses utiles

- **:!**`commande` lance `commande` dans le shell et revient á VIm une fois terminée. Ca peut être utile pour voir le contenu du dossier courant **:!ls**, compiler un programme **:!gcc *.c -o projetC**
- **:help *commande*** renvoie l'aide de VIm pour *commande*.
- **:**[<**Up**>],**:**[<**Down**>] navigue dans les commandes précédemment tapées.
- [**.**] refait la dernière commande.
- **:ab *mot1 mot2*** après cette commande lorsqu'on tapera *mot1* [**SPC**],",",". " il sera transformé en *mot2*
  exemple: Tout au long de ce document les trois abbréviations **:ab bitem beginitemize**, **:ab eitem enditemize** et **:ab item**
- m'ont été très utiles. Il semble cependant que cela ne marche que sur gVIm donc tester chez vous...
- [<**Ctrl**>**G**] Indique la position dans le fichier du curseur.
- **:*x,y*g/*str*/*cmd*** Éxecute *cmd* sur chaque occurence de *str* trouvée dans l'intervalle indiqué
- [<**Ctrl**>**a**,<**Ctrl**>**x**] incrémente ou décrémente le chiffre sous le curseur.

5 Les fichiers

5.1 Ouvrir un fichier

Executer la commande `vim` *`fichier1 fichier2`* dans votre terminal pour lancer VIm en edition de fichier on passera au fichier suivant avec **:n** au précédent avec **:N**. On peut lancer VIm sans argument puis une fois en mode commande taper **:e *fichier***. La commande **:tabnew *fichier*** lancera fichier dans un nouvel onglet et on passera d'un onglet à l'autre avec **[Ctrl][PgUp]**, **[Ctrl][PgDn]**.

5.2 Fermer et sauver un fichier

- **:w *fichier*** Si fichier est absent alors la commande enregistre le fichier édité sur lui-même, si fichier est présent le fichier édité est sauvé sous son nouveau nom qui sera crée si il n'existe pas.
- **:wq,[ZZ],:x** Ces trois commandes sauvent et quittent le fichier en cours, faire attention si le fichier édité est en lecture seule (ce qui peut arriver si il ne vous appartient pas, cf projet C) je ne garantis pas le comportement il est possible qu'il ne sauve rien mais qu'il quitte quand même.
- **:q** Quitte le fichier en cours. VIm ne vous autorisera á faire cela que si les dernières modifications ont été enregistrées.
- **:q!** Quitte le fichier en cours sans condition.

5.3 Autre

- **:r *fichier*** insere le contenu de *fichier* dans le fichier édité á la position courante du curseur.

6 Mouvements dans un fichier

Les opération de mouvement sont particulièrement puissantes lorsqu'elles sont couplées aux commandes d'éditions copier, coller ou supprimer. Certaines ne peuvent pas être couplées, cela sera précisé.

- **[w]** avance d'un mot.
- **[b]** recule d'un mot.
- **[$]** place le curseur á la fin de la ligne.
- **[0]** place le curseur au début de la ligne.
- [^] place le curseur sur le premier caractère de la ligne.

- [)] avance d'une phrase (une suite de mots se terminant par un point).
- [(] recule d'une phrase.
- [**}**] avance d'un paragraphe (une suite de phrase suivie d'une ligne vide).
- [**{**] recule d'un paragraphe.
- ***:n*** place le curseur au début de la ligne *n*. elle n'est pas couplable (á ma connaissance).
- [%] si le curseur est placé sur une parenthêse une accolade ou un crochet alors a renvoie la parenthêse accolade ou crochet correspondant.
- [**h**] á gauche d'un caractère.
- [**l**] á droite d'un caractère.
- [**j**] descend d'une ligne.
- [**k**] monte d'une ligne.
- ***/str*** place le curseur sur la prochaine occurence de *str*.
- ***?str*** place le curseur sur la prćédente occurence de *str*.
- [**f*x***] place le curseur sur la prochaine occurence du caractère x sur la ligne courante.
- [**F*x***] place le curseur sur la précédente occurence du caractère x sur la ligne courante.
- [**t*x***] place le curseur juste avant la prochaine occurence du caractère x sur la ligne courante.
- [**m*lettre***] enregistre la position du curseur dans la lettre on y retourne en tapant [**'*lettre***].Notez que Vim fait la différence entre les majuscules et minuscules. Très utile lorsqu'on édite un gros fichierou encore quand on a identifié avec précision l'endroit exact á partir duquel on a écrit de la merde.
- [*] recherche la prochaîne occurence du mot sous le curseur.
- [#] recherche l'occurence précédente du mot sous le curseur.

7 Selection de texte

La sélection comme les mouvements peut être couplée aux opérations d'edition.

- [**v**] sélectionne une zone á partir du curseur courant.
- [**V**] sélectionne une zone á partir du début de la ligne courante.
- [<**Ctrl**>**v**] sélectionne une zone rectangulaire á partir du curseur. Ce n'est pas évident á première vue mais c'est une commande pratique. exemple:

```
i=1+2;                        i=1-2;
j=3+4;---[<Ctrl>v][r-]---> j=3-4;
k=5+6;                        k=5-6;
```

8 Operations

Ces opérations peuvent être autonomes, si elles sont préc'ed'es d'une selection elles agiront sur la selection et si elles sont suivies d'un mouvement elles agiront sur ce dernier.

8.1 Suppression

- [***n*dd**] supprime n lignes en comprenant celles sur laquelle est le curseur.
- [***n*D**] supprime n lignes á partir du curseur.
- ***n*x** supprime n caractères á partir de la position du curseur.
- ***n*d*mvt*** supprime n fois selon le mouvement *mvt* qui suit.
- ***sel*d** supprime la sélection.

exemple:

- [**5dd**] supprime les 5 lignes d'après.
- [**5dw**] supprime les 5 prochains mots.
- [**d$**] supprime jusqu'á la fin de la ligne.
- [**d0**] supprime jusqu'au début de la ligne.
- [**d}**] supprime jusqu'á la fin du paragraphe.
- [**3dj**]supprime les 3 prochaines lignes.
- [**d**]/**grenoble** supprime tout jusqu'á la prochaine occurence de grenoble.

- **[dfx]** supprime tout les caractères sur la ligne jusqu'á x compris.
- **[d%]** supprime tout jusqu'á la parenthèse, le crochet ou l'accolade correspondante.
- **:g/^$/d** supprime toutes les lignes vide dans un fichier.

8.2 Copier

- **[*n*yy]** copie *n* lignes en comprenant celles sur laquelle est le curseur.
- ***n*y*mvt*** copie *n* fois selon le mouvement *mvt* qui suit.
- ***sel*y** copie la sélection.

exemple:

- **5yy** copie les 5 lignes d'après.
- **5yw** copie les 5 prochains mots.
- **y$** copie jusqu'á la fin de la ligne.
- **y0** copie jusqu'au début de la ligne.
- **y}** copie jusqu'á la fin du paragraphe.
- **3yj**copie les 3 prochaines lignes.
- **y/grenoble** copie tout jusqu'á la prochaine occurence de grenoble.
- **yfx** copie tout les caractères sur la ligne jusqu'á x compris.
- **y%** copie tout jusqu'á la parenthèse, le crochet ou l'accolade correspondante.
- **y'm** copie tout entre le curseur courant et la marque.

8.3 Coller

- **[p]** colle le texte après le curseur.
- **[P]** colle le texte avant le curseur.

8.4 Substitution

- **:*x,y*s/*str1*/*str2*/*cg*** remplace le texte *str1* par *str2* dans l'intervalle désigné. *c* signifie que vous voulez confirmer chaque changement et *g* signifie que le changement sera fait plusieurs fois par ligne et non une seule fois. Ils sont bien sur optionnels.

exemple:

- **%s/toto/tata/g**remplace toto par tata dans tout le fichier.
- **10,20s/toto/tata/** remplace la première occurence de toto par tata entre les lignes 10 et 20.
- **%/^$/ligne vide/** remplace toute les lignes vides par ligne vide.
- **%/ŷep/yop** remplace toutes les yep qui commence une ligne par des yop.

9 Commande

9.1 map

On peut associer une suite de commande á une seule touche ici rien de mieux qu'un exemple. Il est assez fastidieux de commenter de nombreuses lignes de code en HTML, en effet pour chaque ligne il faudrait

- aller en début de ligne.
- passer en mode insertion et rajouter <!–.
- aller á la fin de la ligne et rajouter –>.

La commande **map** <**F2**> **0i**<**!**–<**ESC**>**$a**–><**ESC**> permet de commenter une ligne entière en appuyant sur F2 en effet le 0 passe au début de ligne i passe en mode insertion où on insère le début de commentaire puis on sort du mode insertion, on passe en fin de ligne, on rajoute la fin de commentaire et on sort du mode insertion. Que demande le peuple?

9.2 macro

Dans le même esprit on peut demander á VIm de littéralement enregistrer les séquences de touche que l'on tape. On commence l'enregistrement par [**q*lettre***], on l'arrête par [**q**], et on rapelle la séquence par [**@*lettre***]. Encore une fois rien ne vaut l'exemple. Si on veut imprimer les valeurs successives de la suite de fibonacci on commence par écrire:

```
printf("fibo(%d)=%d",0,fibo(0));
```

Puis on commence l'enregistrement par q*lettre* on utilise l'incrémentation de variable

10 VIm pour le programmeur

10.1 Folding

Le folding désigne le fait de plier et déplier á volonté des bouts. Je l'ai mis dans la section programmation car c'est pratique de pouvoir observer l'ensemble d'un code 1000 lignes d'un seul coup oeil et aussi d'observer juste la première et la dernière fonction dans un seul écran. Notez que les folds peuvent être imbriqués selon un système de niveau.

- [***sel*zf**] crée un fold constitué de la sélection précédente.
- [**zo**] ouvrir le fold sous le curseur.
- [**zj**] déplace le curseur jusqu'au prochain fold.
- [**zk**] déplace le curseur jusqu'au fold précédent.
- [**zm**] ferme le fold ouvert sous le curseur.
- [**zM**] ferme tous les folds ouverts.
- [**zd**] supprime le fold sous le curseur (ça ne supprime que la structure pas le contenu).
- [**zE**] supprime tous les folds.

10.2 Initialisation

Dans un code source si on tape **gd** sur une variable le curseur se positionne sur la déclaration de cette variable.

10.3 Documentation

En tapant **K** sur une commande connue on lance sa page de manuel.

11 Sources

- lea-linux
- Digital-Fashion
- VI lovers
- CommentCaMarche
- vim.org

<++>