Vim
Text processing at maximum speed and power

17 décembre 2008

Text editor vs word processor

Vi est un text editor (un autre exemple est EMACS).

Vi n'est pas un word processor comme Word Office, Open office Writer, Kwrite, ...
Différence entre WYSIWYG (What You See Is What You Get) et WYSIWYM (What You See Is What You Mean).

Un peu d'histoire

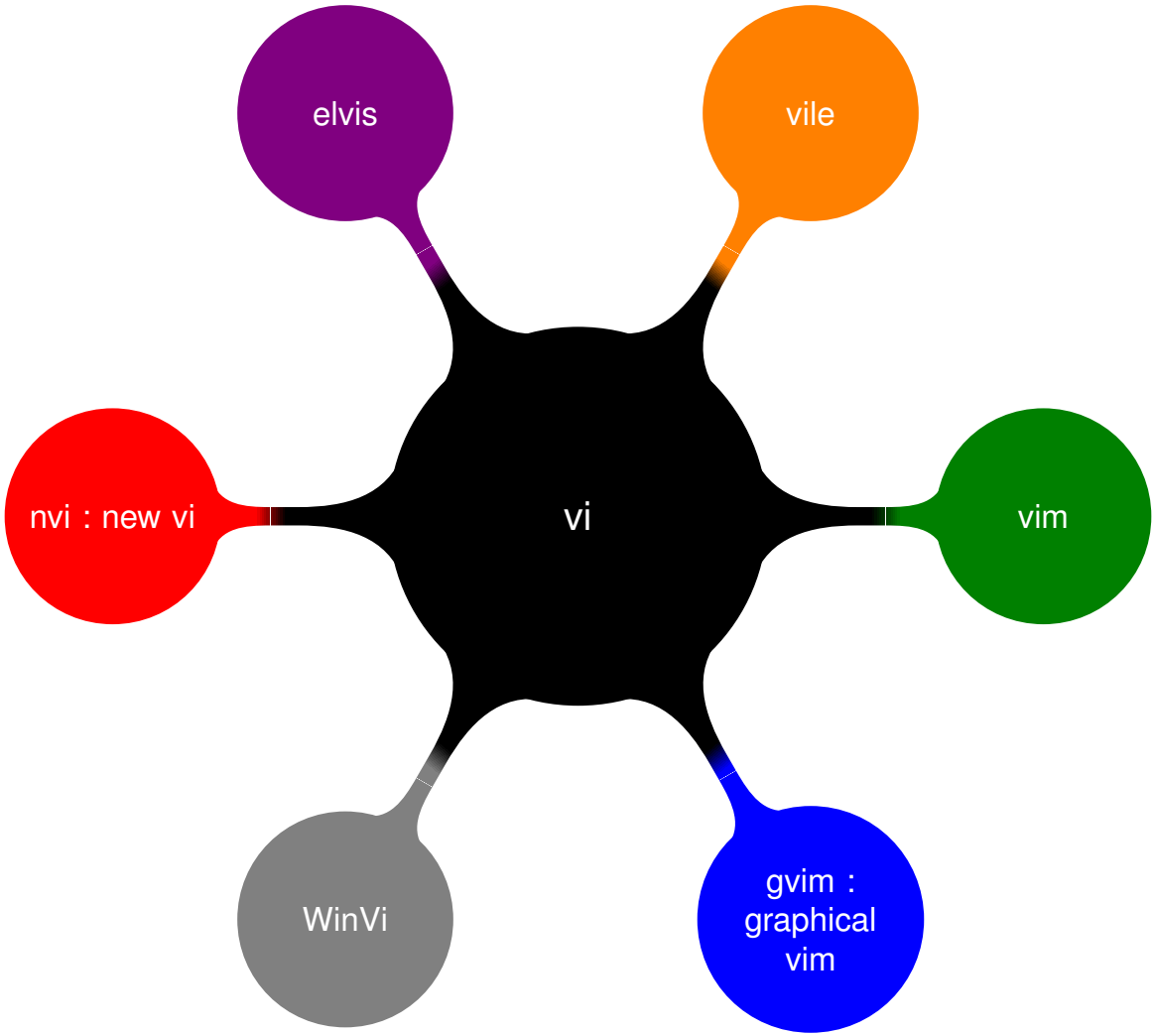
Vi a été écrit en 1976 par Bill Joy sous BSD :

- Le nom vient de l'abréviation de visual.

ViM est apparu en 1991 par Bram Moolenaar :

- Signifie "VI IMproved" ou "VI Meilleur" ;
- Vim est actuellement à la version 7.1.

Vi et ses clones



Quel est l'interet ?

« VI VI VI, le chiffre de la bête »

Avantages :

- disponible sous tous les systèmes UNIX-like (indispensable dans des cas d'utilisation avancée du système) ;
- rapidité, les commandes sont à une ou deux touches ;
- très léger ;
- exécution batch mode (les opérations peuvent être très compliquées, i.e. regular expression).

Désavantages :

- non user-friendly ;
- temps d'apprentissage obligatoire.

Les différents modes

- Mode Normal (Normal Mode) :
  - mode dans lequel on arrive lorsqu'on lance ViM
  - ESC pour y revenir
- Mode Insertion (Insert Mode) :
  - comme son nom l'indique, il sert à insérer du texte
  - i pour accéder à ce mode
- Mode Visuel (Visual Mode) :
  - pour sélectionner du texte
  - v pour y entrer
- Mode Ligne-de-Commande (CommandLine Mode) :
  - pour faire tout un ensemble d'actions entrées par des commandes
  - : pour y entrer

Déplacement

- h, j, k, l pour se déplacer (h a gauche, l a droite, j en bas, k en haut)
  Les flèches marchent, mais se servir de hjkl permet de gagner beaucoup de temps ? !
- G pour aller à la fin du fichier
- gg pour aller au début
- ^ pour aller au début de la ligne
- $ pour aller à la fin de la ligne

Insertion et modification de texte

- i pour insérer du texte avant le curseur
- a pour insérer du texte après le curseur

- I pour insérer du texte au début de la ligne (premier caractère imprimable)
- A pour insérer du texte à la fin de la ligne

- o pour insérer du texte en dessous de la ligne courante
- O pour insérer du texte au dessus de la ligne courante

- dd supprime la ligne courante, 5dd supprime 5 lignes
- D supprime jusqu'à la fin de la ligne

Copier/coller et undo/redo

Toute suppression de texte est automatiquement copiée dans le registre par défaut

yy pour copier une ligne (yank)
p et P pour coller le registre par défaut après et avant le curseur (paste)

"ayy copier la ligne dans le registre a
"ap coller le registre a

u pour annuler
Ctrl r pour refaire

Le mode commande

Dans le mode normal, : fait entrer dans le mode commande.
Les commandes sont donc précédées d'un : mais pas forcément.
Pour sortir du mode commande :

- CR exécute la commande, et revient dans le mode normal
- sinon ESC pour revenir dans le mode normal

Ouvrir, enregistrer, quitter

- :e fichier.txt ouvrir le fichier «fichier.txt»
- :w enregistrer le fichier courant
- :w fichier.txt enregistrer sous le nom «fichier.txt».
- :q quitter
- :wq enregistrer et quitter.

Find/replace

On peut utiliser les “regular expression”.
La recherche est très simple :
/mot_cherché n et N pour le prochain et pour le précedent.
Le remplacement des mots...très compliqué ? !
:s/old/new/

- `:%s/o.d/odd/g`
- `:%s/o.d/odd/g`
- `:%s/^old/new/gc`
- `:1,10s/.*/(&)/`
- `:%s/\(that\) or \(this\)/\2 or \1/g`

Remplacement “context sensitive” :g/pattern/s/old/new/options

- `:g/editeur/s/editeur/editor/g`
- `:g/includegraphics/s/eps/pdf/g`

Abbreviation et mapping

:ab abbr phrase
exemple :

```
:ab ;m lnalod@locean-ipsl.upmc.fr
```

:map x sequence
on definit le caractère x comme une séquence de commandes d'écriture.
exemples :

```
:map v lbi{\italic ^[e}^[
```

pour transformer en italique un mot en LATEX : {\textit mot}

```
:map v dwelp
```

pour renverser deux mots

Programmation

- indentation automatique : =, >, <
- fenêtres multiples :
  - :sp, :vsp horizontal and vertical split
  - *n*z n=4 –> 4z
  - Ctrl w flèches pour se deplacer
- folding lines :
  - zf*n*j n=2 –> zf2j pour faire le fold de 2 lignes
  - za pour faire le fold/defold
  - zr pour faire defold récursif
- autocompletion ...

Autocompletion

- mots –> CtrlX CtrlN
- ligne –> CtrlX CtrlL
- file –> CtrlX CtrlF
- classe –> CtrlX CtrlO (il faut faire ctags et ajouter le plugin au préalable !)

Divers

- ajouter un fichier dans un fichier

```
:r fichier.txt
```

- ou la date

```
:r !date
```

- contrôle d'ortographe

```
:!aspell -l en -c %
```

- sorting

```
:1,3 !sort
```

- marks

```
ma  pour marquer et  'a  retourner
```

Le jeu en vaut-il la chandelle ?

- pas que pour les *g*eeks

- cela vaut peut-être la peine d'investir du temps une fois pour toutes sur quelque chose qui sera sûrement utile pour toute occasion
- on n'est pas obligé d'apprendre tout d'un seul coup

Lectures complémentaires

- Learning the Vi and Vim editors 7th edition.
  Arnold Robbins, Elbert Hannah and Linda Lamb.
  O'REILLY.
- Documentation complète vim.
  http ://www.vim.org/.
- Les scripts.
  http ://www.vim.org/scripts .

YAO home page : *http://www.locean-ipsl.upmc.fr/∼lnalod*
YAO groupe email : *lnalod@locean-ipsl.upmc.fr*

Questions