

VINCENT JOUSSE

VIM POUR LES HUMAINS

VINCENT JOUSSE

Copyright © 2013 Vincent Jousse

PUBLIÉ PAR VINCENT JOUSSE

STYLE L^AT_EX <http://tufte-latex.googlecode.com>

Copie de 03 avril 2013 destinée à la relecture uniquement

Table des matières

<i>Rendre Vim utilisable</i>	17
<i>L'outil de manipulation de texte rêvé</i>	33
<i>Les plugins indispensables</i>	47
<i>Pense-bête et exemples</i>	57

Table des figures

1	Le thème <i>Solarized</i> en sombre et en clair. http://ethanschoonover.com/solarized	17
2	Nouveau fichier vide.	18
3	Mon premier commentaire.	19
4	Position de repos, clavier QWERTY. Illustration par Cy21 - CC-BY-SA-3.0 (www.creativecommons.org/licenses/by-sa/3.0) ou GFDL (www.gnu.org/copyleft/fdl.html), via Wikimedia Commons http://commons.wikimedia.org/wiki/File:Typing-home-keys-hand-position.svg	20
5	<i>Vim</i> après votre première configuration.	23
6	Coloration syntaxique par défaut.	24
7	Le contenu du répertoire <code>.vim</code> avec <i>Solarized</i> .	24
8	Le thème <i>Solarized</i> sombre.	25
9	Le thème <i>solarized</i> clair.	25
10	<code>.vim</code> avant et après Pathogen.	27
11	<code>.vim</code> avec <i>The NERD Tree</i> d'activé.	30
12	<i>Vim</i> en mode insertion.	35
13	<i>Vim</i> prêt pour le copier/coller.	35
14	<i>Vim</i> après le copier/coller.	36
15	Les « touches directionnelles » de <i>Vim</i> en mode normal.	37
16	Les touches <i>Vim</i> .	39
17	Le clavier sur lequel <i>Vi</i> a été réalisé.	40
18	<i>Vim</i> adventures, une façon ludique d'apprendre <i>Vim</i> .	45
19	<i>Vim</i> avec Lusty Explorer d'activé en bas.	49
20	Lusty Explorer et le Fuzzy matching.	50
21	<i>Vim</i> avec la fenêtre Quickfix de Ack.	52
22	<i>Vim</i> avec CtrlP de lancé.	55

Liste des tableaux

*Merci à Brett Kelly (rédacteur d'Evernote
Essentials) pour avoir pris le temps de me
guider.*

Introduction

LORSQUE LE BESOIN d'écrire ou de coder se fait se sentir, le choix d'un éditeur de texte est primordial. Il en existe énormément sur le "marché", mais peu d'entre eux peuvent se targuer d'environ 40 ans d'existence. C'est le cas d'*Emacs*¹, de *Vi* et de son "successeur" *Vim*². Ils ont été créés dans les années 70 et sont toujours très utilisés actuellement³. Comme vous avez sans doute pu le voir, ce n'est pas grâce à la beauté de leur site internet ou à l'efficacité de leur communication. Voici quelques **raisons de leur succès** :

Pour la vie

Ils s'apprennent une fois et s'utilisent pour toujours. Dans un monde où les technologies/langages changent tout le temps, c'est une aubaine de pouvoir investir sur du long terme.

Partout

Ils sont disponibles sur toutes les plateformes possibles et imaginables (et l'ont toujours été).

Augmentent votre productivité

Ils présentent un rapport temps investi / gain de productivité fabuleux.

Couteaux Suisses

Ils permettent d'éditer tout et n'importe quoi. Quand vous changez de langage de programmation, vous n'aurez pas à changer d'éditeur. À noter que ce livre a bien sûr été écrit avec *Vim*.

Et pourtant, ils restent difficiles à apprendre. Non pas qu'ils soient plus compliqués qu'autre chose, non pas que vous ne soyez pas à la hauteur, mais plutôt à cause d'un manque de pédagogie des différentes documentations.

Ce livre a pour but de pallier ce manque en vous guidant tout au long de votre découverte de *Vim*⁴. Il ne prétend pas être un guide exhaustif⁵ mais plutôt un bon moyen de gagner du temps en allant à l'essentiel et en vous évitant de parcourir tout internet pour trouver le moyen le plus rapide de tirer parti de *Vim*.

1. <http://www.gnu.org/software/emacs/>

2. <http://www.vim.org/>

3. À noter que *Vim* n'est arrivé qu'en 1991

4. Je laisse *Emacs* à ceux qui savent. Pour un bref comparatif c'est ici : http://fr.wikipedia.org/wiki/Guerre_d'éditeurs. Les goûts et les couleurs ...

5. Vous pouvez essayer *A Byte of Vim* pour cela : <http://www.swaroopch.org/notes/Vim>

Vous aussi vous en avez marre d'attendre la release de TextMate 2⁶ ? D'essayer le n-ième éditeur à la mode et de devoir tout réapprendre et ce jusqu'à la prochaine mode ? De devoir changer d'éditeur quand vous passez de votre Mac, à votre Windows, à votre Linux ? Alors vous aussi, rejoignez la communauté des gens heureux de leur éditeur de texte. **Le changement, c'est maintenant.**

6. À noter que depuis l'écriture de ce livre, le code de TextMate 2 a été publié sous licence GPL : <https://github.com/textmate/textmate>

Pour qui ?

TOUTE PERSONNE étant amenée à produire du texte (code, livre, rapports, présentations, ...) de manière régulière. Les développeurs sont bien sûr une cible privilégiée, mais pas uniquement.

Par exemple vous êtes :

Étudiant Si vous voulez faire bien sûr un CV, c'est un must (en plus d'être un attrape geekette en puissance⁷). Vous aurez un outil unique pour écrire tout ce que vous avez à écrire (et que vous pourrez réutiliser tout au long de votre carrière) : vos rapports en L^AT_EX, vos présentations⁸, votre code (si vous avez besoin d'OpenOffice ou de Word pour écrire vos rapports, il est temps de changer d'outil et d'utiliser L^AT_EX).

7. À confirmer.

8. En utilisant *impress.js* par exemple : <http://bartaz.github.com/impress.js>. Basé sur du HTML/JS/CSS, je vous le recommande grandement pour des présentations originales et basées sur des technologies non propriétaires.

Enseignant Il est temps de montrer l'exemple et d'apprendre à vos étudiants à bien utiliser un des outils qui leur servira à vie, bien plus qu'un quelconque langage de programmation.

Codeur Investir dans votre outil de tous les jours est indispensable. Quitte à apprendre des raccourcis claviers, autant le faire de manière utile. Si cet investissement est encore rentable dans 10 ans, c'est juste l'investissement parfait, c'est *Vim*.

Administrateur système Unix Si vous utilisez *Emacs* vous êtes pardonnable, si vous utilisez nano/pico je ne peux plus rien pour vous, sinon il est grand temps de s'y mettre les gars, c'est un des cas d'utilisation parfait (un éditeur de texte surpuissant ne nécessitant pas d'interface graphique).

Écrivain Si vous écrivez en markdown/RST/WikiMarkup ou en L^AT_EX, *Vim* vous fera gagner beaucoup de temps. Vous ne pourrez plus repasser à un autre éditeur, ou vous voudrez le "Vimifier" à tout prix.

Faites moi confiance, je suis passé et repassé par ces 5 rôles, mon meilleur investissement a toujours été *Vim*, et de loin.

Ce que vous apprendrez (entre autres choses)

- Comment utiliser *Vim* comme un éditeur "normal" d'abord (vous savez, ceux qui permettent d'ouvrir des fichiers, de cliquer avec la souris, qui ont une coloration syntaxique ...). En somme, la démystification de *Vim* qui vous permettra d'aller plus loin.
- Comment passer de l'édition de texte classique à la puissance de *Vim*, petit à petit (c'est là que l'addiction commence).
- Comment vous passer de la souris et pourquoi c'est la meilleure chose qu'il puisse vous arriver quand vous programmez/tapez du texte.
- Comment vous pouvez facilement déduire les "raccourcis claviers" avec quelques règles simples.

Si je devais le résumer en une phrase : puisque vous vous considérez comme **un artiste, passez du temps à apprendre** comment utiliser l'outil qui vous permet de vous exprimer, une bonne fois pour toute.

Ce que vous n'apprendrez pas (entre autres choses)

- Vous n'apprendrez pas comment installer/configurer *Vim* pour Windows. Pas que ce ne soit pas faisable, mais je n'ai que très peu de connaissances sous Windows. Ça viendra peut-être, mais pas tout de suite. On couvrira ici Linux/Unix (et par extension Mac Os X).
- Vous n'apprendrez pas comment utiliser *Vi* (notez l'absence du "m"). Je vais vous apprendre à être productif pour coder/produire du texte avec *Vim*, pas à faire le beau devant les copains avec *Vi*⁹. Pour ceux qui ne suivent pas, *Vi* est "l'ancêtre de *Vim* (qui veut dire *Vi* - *IMproved*, *Vi* amélioré)" et est installé par défaut sur tous les Unix (même sur votre Mac OS X).
- Vous n'apprendrez pas à connaître les entrailles de *Vim* par cœur : ce n'est pas une référence, mais un guide utile et pragmatique.
- Vous n'apprendrez pas comment modifier votre *Vim* parce que vous préférez le rouge au bleu : je vous ferai utiliser le thème *Solarized* (<http://ethanschoonover.com/solarized>), il est juste parfait pour travailler.

9. *Vim* est suffisant pour cela de toute façon.

Le plus dur, c'est de commencer

Alors, prêt pour l'aventure ? Prêt à sacrifier une heure pour débiter avec *Vim*, une semaine pour devenir familier avec la bête, et le reste de votre vie pour vous féliciter de votre choix ? Alors c'est parti !

Enfin presque, il faut qu'on parle avant.

Vim fait partie de ces outils avec lesquels vous allez galérer au début. Le but de ce guide est de vous mettre le pied à l'étrier et de diminuer la hauteur de la marche à franchir. Mais soyez conscients que vous mettre à *Vim* va vous demander de la volonté et quelques efforts. Comme on dit souvent, on n'a rien sans rien. Voici la méthode que je vous recommande pour apprivoiser la bête :

- Ne l'utilisez pas comme principal outil de travail au début.
Faites des séances de 20/30 minutes par jour où vous vous forcerez à l'utiliser. Ce qui est important c'est de faire entrer *Vim* dans vos habitudes. Si vous le faites trop brusquement il y a de grandes chances que vous abandonniez au bout de quelques jours. Lorsque vous vous sentirez à l'aise avec les bases vous pourrez commencer à l'intégrer dans votre travail quotidien.
- Gardez une feuille avec les principaux raccourcis imprimée à côté de vous. Comme tous les raccourcis claviers il va d'abord falloir les mémoriser, et il n'y a pas mieux qu'un pense-bête et un peu de pratique.
- Gardez la foi. Au début vous ne comprendrez pas bien pourquoi vous devez passer du temps à apprendre un truc qui vous fait tant perdre de productivité. Et puis un jour vous aurez un déclic et vous vous demanderez pourquoi tous vos logiciels ne peuvent pas se contrôler avec les commandes de *Vim*.
- Gardez à l'esprit que c'est un investissement pour vos 20 prochaines années, et c'est bien connu, un investissement ce n'est pas rentable de suite.

Trêve de bavardage, passons aux choses sérieuses. Go go go !

Rendre Vim utilisable

ÇA PEUT PARAÎTRE ÉTONNANT comme approche, mais c'est pour moi la première chose à faire : rendre *Vim* utilisable par un humain lambda. Si tout le monde semble s'accorder sur le fait que *Vim* est un **éditeur très puissant**, tout le monde pourra aussi s'accorder sur le fait que de base, il est juste **imbitable**. Soyons honnête, sans une configuration par défaut minimale, utiliser *Vim* est **contre-productif**.

C'est à mon avis le premier obstacle à surmonter avant toute autre chose. C'est ce que les autres éditeurs « à la mode » comme Textmate, Sublimetext, Notepad++ ou Netbeans proposent, c'est à dire un environnement à minima utilisable tel quel, même si l'on en n'exploite pas la totalité.

Voici donc ce qui manque à un *Vim* nu (et ce qui est pour moi une **cause d'abandon pour beaucoup** d'entre vous) :

Configuration par défaut *Vim* est configurable grâce à un fichier nommé *.vimrc*, qui est bien entendu vide par défaut. La première étape va être d'avoir un fichier *.vimrc* avec une configuration minimale.

Coloration syntaxique De base, *Vim* est tout blanc et tout moche.

On va utiliser le thème *Solarized* (<http://ethanschoonover.com/solarized>). Si votre but est d'être efficace, c'est le meilleur thème disponible actuellement (tout éditeur de texte confondu). La figure 1 vous donne une idée des deux looks disponibles (clair ou sombre). Pour ma part j'utilise le thème sombre.

Explorateur de fichiers Si vous utilisez *Vim* avec une interface graphique (ce qui est le cas de 99% d'entre vous je suppose) vous avez par défaut un menu Fichier vous permettant d'ouvrir un fichier. C'est certes un bon début, mais avoir à disposition un explorateur de projet à la Netbeans ou à la Textmate peut s'avérer très pratique. Pour obtenir le même comportement, nous utiliserons *Nerd-tree* (http://www.vim.org/scripts/script.php?script_id=1658). À savoir qu'à la fin de ce guide, vous n'aurez plus besoin de la souris (et donc des menus et autres boutons).

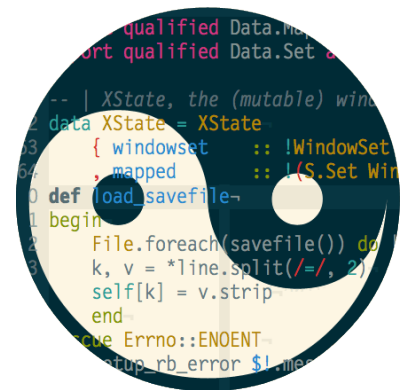


FIGURE 1: Le thème *Solarized* en sombre et en clair. <http://ethanschoonover.com/solarized>

Ce chapitre est indispensable si vous n’avez que peu d’expérience (voire pas du tout) avec *Vim*. À la fin de ce chapitre, vous aurez un *Vim* dont vous pourrez commencer à vous servir pour vos tâches de tous les jours. Cela devrait être suffisant pour vous permettre d’apprendre le reste petit à petit. Car il n’y a pas de secret, il vous faudra pratiquer pour apprendre *Vim*, alors autant commencer de suite et le moins douloureusement possible.

En revanche, si vous êtes déjà familier avec *Vim* et n’utilisez déjà plus la souris, vous pouvez sagement sauter ce chapitre (soyez sûr tout de même de donner sa chance au thème *Solarized*).

Préambule indispensable : le mode insertion

Prenons le pari de créer le fichier `.vimrc` avec *Vim* lui même. Comme je vous le disais, le plus tôt vous commencerez, le mieux ce sera. Vous devrez certainement commencer par installer une version de *Vim*. Si vous utilisez un Mac, essayez MacVim¹⁰ sans aucune hésitation. Si vous utilisez GNU/Linux ou tout autre système “Unix” vous devriez sûrement avoir gVim à votre disposition (ou tout du moins facilement installable grâce à votre gestionnaire de logiciels). Pour Windows, il semblerait y avoir une version disponible sur le site officiel de *Vim*¹¹, mais je ne l’ai pas testée.

Cliquez sur Fichier (File) -> Nouveau (New). Le texte d’accueil par défaut de *Vim* devrait avoir disparu et vous devriez avoir une page blanche comme sur la figure 2.

10. MacVim : <http://code.google.com/p/macvim/>

11. Page de téléchargement officielle de *Vim* : <http://www.vim.org/download.php>

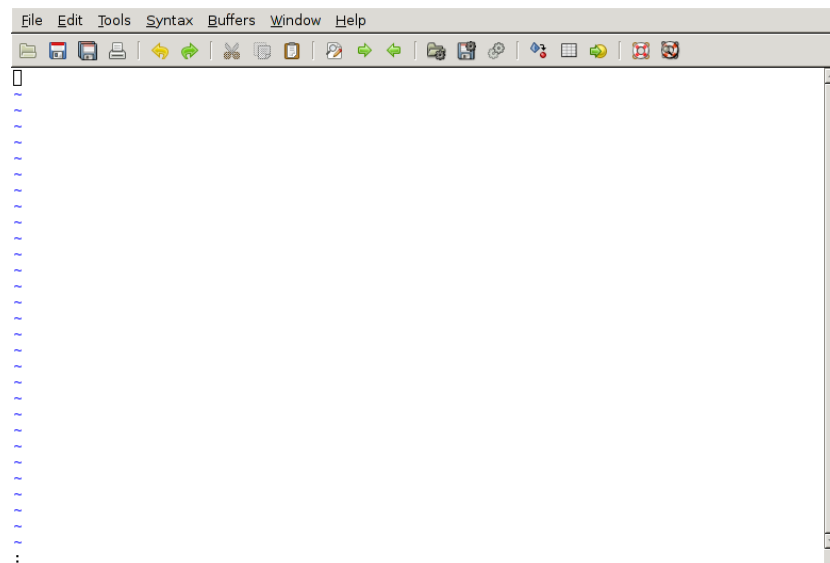


FIGURE 2: Nouveau fichier vide.

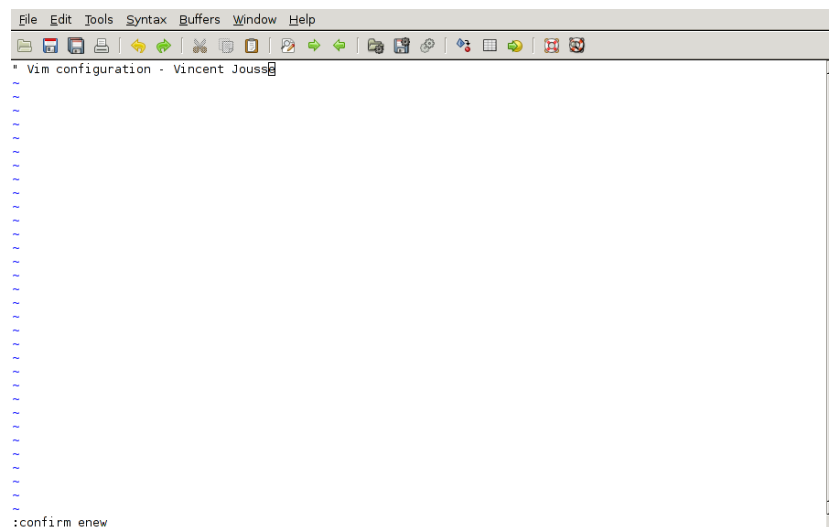
Commençons par entrer un commentaire dans l’en-tête du fichier

pour y mentionner notre nom. Pour pouvoir entrer du texte appuyez sur la touche **i** (le curseur devrait changer d'aspect) et entrez le commentaire ci-dessous ¹².

```
" VIM Configuration - Vincent Jousse
```

Listing 1: Votre première ligne avec *Vim*.

Vous aurez remarqué que les commentaires en *VimL* (le langage de configuration de *Vim*) commencent par un `"`. Appuyez ensuite sur la touche **Esc** (**Échap**) pour revenir au mode par défaut (le mode normal) de *Vim*. Et voilà le travail, cf figure 3.



12. Si vous ne savez pas trop ce que vous avez fait et que *Vim* vous affiche des trucs en rouge en bas à gauche au ne semble pas réagir comme il faut quand vous appuyez sur la touche **i**, appuyez plusieurs fois sur la touche **Esc** (**Échap**), ça devrait vous remettre au mode par défaut de *Vim*

FIGURE 3: Mon premier commentaire.

Tout ça pour ça me direz-vous, et vous avez bien raison. Mais tout cela a une logique que je vais vous expliquer. L'avantage de *Vim* est qu'il est généralement logique. Quand vous avez compris la logique, tout vous semble limpide et tomber sous le sens.

Par défaut, *Vim* est lancé dans un mode que l'on appelle le mode "Normal". C'est à dire que ce mode n'est pas fait pour écrire du texte (ça, ça sera le mode "Insert") mais juste pour se déplacer et manipuler du texte. C'est la présence de ces 2 différents modes (il y en a d'autres mais ce n'est pas le sujet pour l'instant) qui fait toute la puissance de *Vim*. Il vous faudra un certain temps pour vous rendre compte de cette puissance par vous-même, alors faites-moi juste confiance pour l'instant.

Si vous vous demandez pourquoi ces modes, pourquoi on semble se compliquer la vie (on se la simplifie en fait) et en quel honneur, dans le mode par défaut, il n'est même pas possible d'insérer du

texte, lisez attentivement la section qui suit.

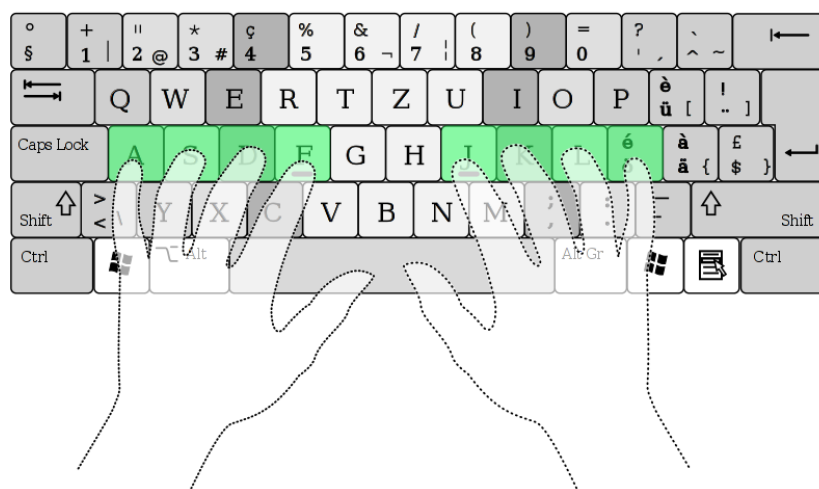
Les modes : d'où Vim tire sa puissance

Je pense que nous serons tous à peu près d'accord sur le fait que si vous souhaitez apprendre à utiliser *Vim*, c'est pour gagner en efficacité pour la saisie/manipulation de texte/code. Pour gagner en efficacité lorsque l'on tape du code il n'y a pas 36 solutions, il n'y en a qu'une en fait : il faut bouger le moins possible les mains (voire pas du tout), et ne bouger que les doigts.

Pour ce faire bien sûr, vous oubliez tout d'abord l'utilisation de la souris. En plus d'être lent, le mouvement clavier -> souris puis souris -> clavier est très mauvais pour vos articulations. Il est souvent à l'origine de troubles musculosquelettiques¹³. D'après *Wikipedia*, c'est le type de maladie professionnelle la plus courante à l'heure actuelle¹⁴.

Vous oubliez aussi le mouvement de votre main droite vers les touches directionnelles gauche/droite/bas/haut. C'est une perte de temps et c'est totalement inutile avec *Vim*.

Qu'est-ce que vous avez le droit de faire dans le coup ? Pas grand chose, si ce n'est garder vos mains sur la position de repos comme le montre la figure 4. Vous trouverez d'ailleurs sur la plus part des claviers des marques sur les touches F et J, c'est pour vous donner un repère tactile de la position où doivent se trouver vos index dans la position de repos.



13. Vous êtes peut-être jeune et n'avez pas encore eu ce type de soucis. Mais croyez moi, ça vient beaucoup plus vite qu'on ne le croit. Si vous passez votre journée sur un ordinateur, ne négligez pas ces facteurs, vous le regretterez un jour.

14. https://fr.wikipedia.org/wiki/Troubles_musculosquelettiques

FIGURE 4: Position de repos, clavier QWERTY. Illustration par Cy21 - CC-BY-SA-3.0 (www.creativecommons.org/licenses/by-sa/3.0) ou GFDL (www.gnu.org/copyleft/fdl.html), via Wikimedia Commons <http://commons.wikimedia.org/wiki/File:Typing-home-keys-hand-position.svg>

Ce parti pris (bouger le moins possible les mains du clavier) justifie à lui seul la présence d'un mode *normal* et d'un mode *insertion*

dans *Vim*. En passant de l'un à l'autre, les touches sous vos doigts serviront tantôt à vous déplacer et à réaliser des opérations sur le texte ¹⁵ (copier/coller, macros, . . .), tantôt à sélectionner ¹⁶ et tantôt à insérer du texte ¹⁷. Tout cela bien sûr en évitant l'utilisation de combinaisons de touches du style *Ctrl + touche* qui ne sont généralement pas bonnes pour vos doigts (*Emacs* si tu nous lis, je te salue).

Par défaut, on passe du mode *insertion* au mode *normal* en appuyant sur la touche **Esc** (**Échap**), mais c'est une des premières choses que l'on changera : la touche **Esc** (**Échap**) est bien trop loin sur les claviers actuels.

Pour passer du mode *normal* au mode *insertion*, on peut par exemple appuyer sur la touche **i**. On apprendra par la suite qu'il existe d'autres moyens de faire. Par exemple pour rentrer en mode insertion tout en créant une nouvelle ligne en dessous de la ligne courante (peu importe où se trouve votre curseur sur la ligne), on utilisera la touche **o** en mode *normal*.

J'y reviendrai plus tard dans «[Se déplacer par l'exemple : Essayer de copier / coller](#)» mais si vous n'êtes pas prêt, à terme, à ne plus utiliser votre souris et les flèches directionnelles pour éditer du texte, je vous recommande presque d'arrêter votre apprentissage maintenant, c'est aussi simple que cela. *Vim* révèle tout sa puissance quand il est utilisé sans souris et en bougeant le moins possible les mains.

SI VOUS VOULEZ POUSSER LA DÉMARCHE encore plus loin, vous pouvez aussi vous procurer un clavier orthogonal *TypeMatrix* ¹⁸. C'est ce que j'utilise personnellement, et mes doigts m'en remercient tous les jours.

L'ultime changement serait d'utiliser une disposition de clavier encore plus efficace comme le *bépo* pour quasi doubler sa vitesse de frappe au clavier. Pour les plus curieux d'entre vous, j'explique la démarche sur mon blog : <http://vincent.jousse.org/comment-doubler-sa-vitesse-de-frappe-au-clavier/>.

15. C'est le mode *normal*

16. C'est le mode *visuel*

17. C'est le mode *insertion*

18. <http://www.typematrix.com/>

La configuration par défaut : indispensable

PASSONS AUX CHOSES SÉRIEUSES, c'est-à-dire comment rendre *Vim* un tant soit peu utilisable. Nous allons donc éditer le fichier de configuration par défaut *.vimrc*¹⁹ en y plaçant des valeurs que toute personne normalement constituée souhaiterait y voir figurer.

J'ai commenté chacune des lignes du fichier directement dans le code. Rien de sorcier ici, on se demande juste pourquoi tout cela n'est pas inclus par défaut.

```
" VIM Configuration - Vincent Jousse
" Annule la compatibilité avec l'ancêtre Vi : totalement indispensable
set nocompatible

" -- Affichage
set title           " Met à jour le titre de votre fenêtre ou de
                    " votre terminal
set number          " Affiche le numéro des lignes
set ruler           " Affiche la position actuelle du curseur
set wrap            " Affiche les lignes trop longues sur plusieurs
                    " lignes

set scrolloff=3     " Affiche un minimum de 3 lignes autour du curseur
                    " (pour le scroll)

" -- Recherche
set ignorecase      " Ignore la casse lors d'une recherche
set smartcase       " Si une recherche contient une majuscule,
                    " re-active la sensibilité à la casse
set incsearch       " Surligne les résultats de recherche pendant la
                    " saisie
set hlsearch        " Surligne les résultats de recherche

" -- Beep
set visualbell      " Empêche Vim de beeper
set noerrorbells    " Empêche Vim de beeper

" Active le comportement 'habituel' de la touche retour en arrière
set backspace=indent,eol,start

" Cache les fichiers lors de l'ouverture d'autres fichiers
set hidden
```

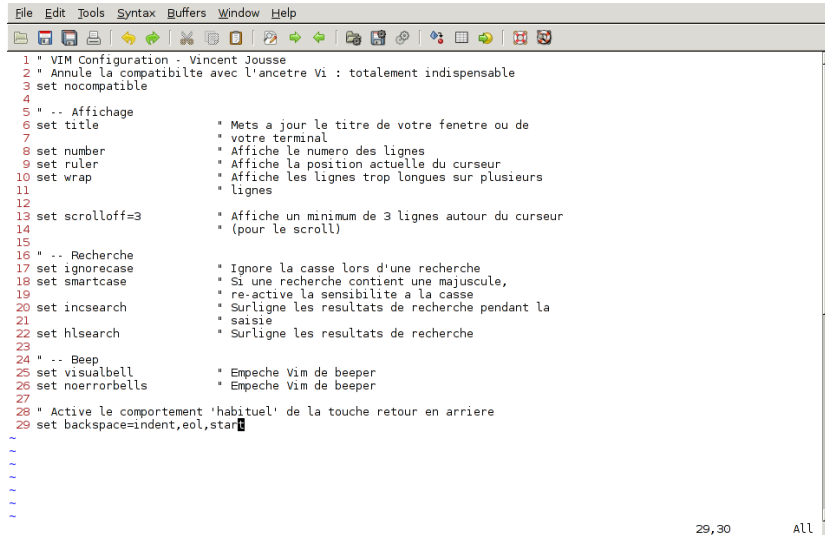
Listing 2: Une configuration par défaut sensée.

J'ai mis en ligne ce fichier de configuration directement sur *Github*. Vous pouvez le télécharger ou le copier directement à partir d'ici : <http://github.com/vjousse/vim-for-humans/fr/firstconfig/>. Il devrait aussi faire partie du package que vous avez téléchargé.

Vous devriez avoir un *Vim* qui ressemble à celui sur la figure 5. Notez les numéros de ligne sur la gauche ainsi que la position du curseur en bas à droite.

Bon c'est bien beau tout ça mais ça manque un peu de couleurs.

19. Ce fichier doit se trouver dans votre répertoire d'accueil. */home/votre_user/.vimrc* sous Linux, */Users/votre_user/.vimrc* sous Mac Os X ou plus généralement *~/.vimrc*. Sous Windows vous pouvez créer un fichier nommé *.vimrc* qui doit se situer dans votre répertoire *%HOME%* qui change en fonction de votre version de Windows. C'est généralement le répertoire juste "au dessus" de votre répertoire *Mes Documents*. Plus d'infos sur Wikipedia http://en.wikipedia.org/wiki/Home_directory#Default_Home_Directory_per_Operating_System

FIGURE 5: *Vim* après votre première configuration.

Au suivant !

Que la couleur soit !

TOUT D'ABORD il faut commencer par activer la coloration syntaxique du code dans le fichier de configuration. Ajoutez ces lignes à la fin de votre fichier de configuration *.vimrc*.

```
" Active la coloration syntaxique
syntax enable
" Active les comportements spécifiques aux types de fichiers comme
" la syntaxe et l'indentation
filetype on
filetype plugin on
filetype indent on
```

Listing 3: Activation de la coloration syntaxique.

Vous devriez avoir un *Vim* qui ressemble à celui de la figure 6²⁰. C'est une bonne première étape, passons maintenant à l'utilisation d'un thème.

Les thèmes vont vous permettre de rendre votre *Vim* un peu moins austère en changeant généralement la couleur de fond ainsi que les couleurs par défaut pour le code. Comme je l'ai mentionné plus haut, nous allons utiliser le thème solarized <http://ethanschoonover.com/solarized> (avec fond clair ou foncé, ça dépendra de vous).

Pour l'installer, commencez tout d'abord par créer un répertoire nommé *.vim*²¹ au même endroit que votre *.vimrc*²². Dans ce réper-

20. Pour l'instant, le plus facile pour que les modifications apportées à votre *.vimrc* soient prises en compte, c'est de le fermer et de le ré-ouvrir. Si vous voulez vraiment vous la jouer à la *Vim* de suite, en mode normal tapez

```
:so ~/.vimrc
:sO
:source
```

:sO étant un raccourci pour

21. Ce répertoire s'appelle *vimfiles* sous Windows. À chaque fois que je ferai référence au répertoire *.vim* ça sera en fait *vimfiles* pour les Windowsiens

22. Dans votre répertoire utilisateur donc.

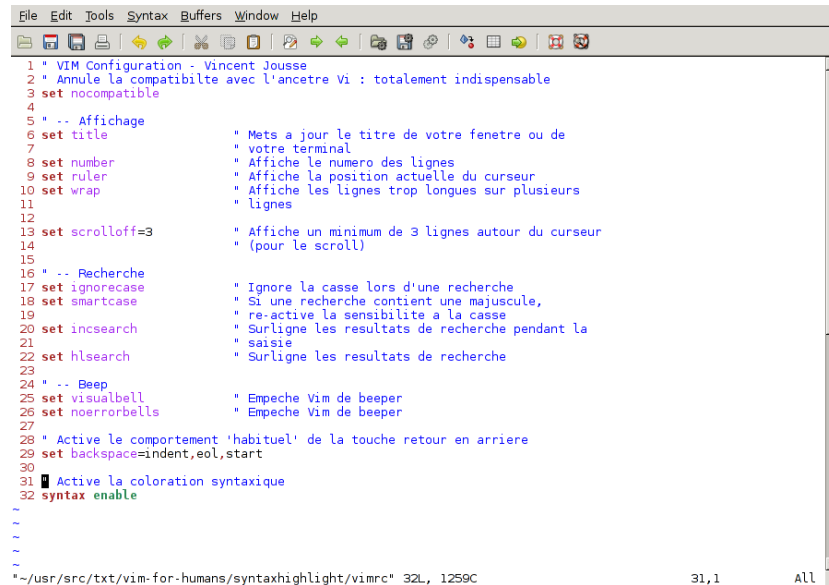
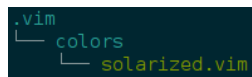


FIGURE 6: Coloration syntaxique par défaut.

toire `.vim`, créez un sous-répertoire nommé `colors`. Téléchargez ensuite le fichier du thème Solarized <https://raw.githubusercontent.com/altercation/vim-colors-solarized/master/colors/solarized.vim>²³ (c'est le même fichier pour les deux versions du thème) et copiez le dans le répertoire `vim/colors/` fraîchement créé. Votre répertoire `.vim` devrait ressembler à celui de la figure 7.

23. ou copiez celui qui vous a été fourni avec le téléchargement de ce livre

FIGURE 7: Le contenu du répertoire `.vim` avec Solarized.

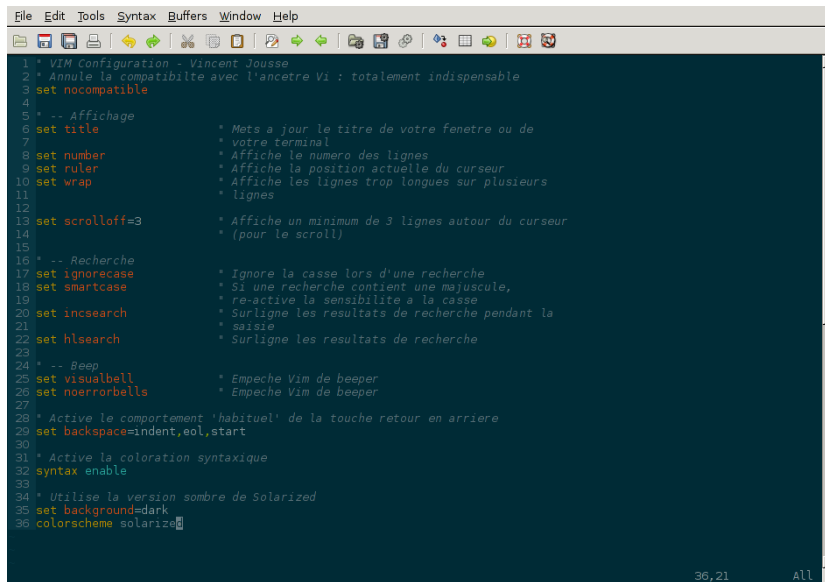
Activez ensuite le thème Solarized dans votre `.vimrc` comme le montre le code dans le listing 4. Pour tester le thème clair, remplacez `dark` par `light` (au niveau de la définition de la propriété `background`).

```
" Utilise la version sombre de Solarized
set background=dark
colorscheme solarized
```

Listing 4: Activation de la coloration syntaxique.

Les images 8 et 9 vous donnent un aperçu des deux variantes (ma préférence allant à la variante sombre soit dit en re-passant).

UN BONUS (si vous n'utilisez pas *Vim* directement dans votre terminal) serait de choisir une police de caractères qui vous convient un peu mieux, c'est bien sûr facultatif mais bon, je présume que certains

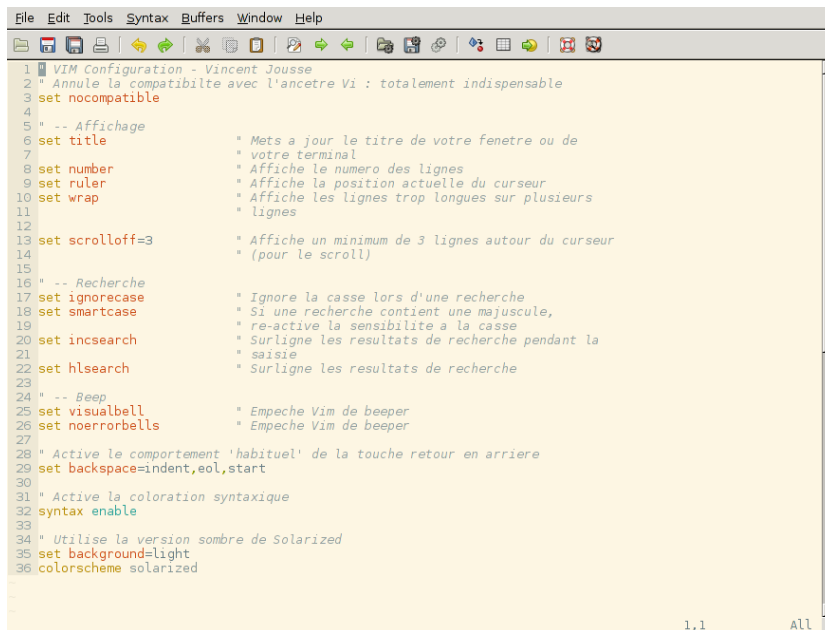


```

1 * VIM Configuration - Vincent Jousse
2 * Annule la compatibilite avec l'ancetre Vi : totalement indispensable
3 set nocompatible
4
5 " -- Affichage
6 set title           " Mets a jour le titre de votre fenetre ou de
7                     " votre terminal
8 set number          " Affiche le numero des lignes
9 set ruler            " Affiche la position actuelle du curseur
10 set wrap            " Affiche les lignes trop longues sur plusieurs
11                    " lignes
12
13 set scrolloff=3      " Affiche un minimum de 3 lignes autour du curseur
14                     " (pour le scroll)
15
16 " -- Recherche
17 set ignorecase       " Ignore la casse lors d'une recherche
18 set smartcase        " Si une recherche contient une majuscule,
19                     " re-active la sensibilit   a la casse
20 set incsearch        " Surligne les resultats de recherche pendant la
21                     " saisie
22 set hlsearch         " Surligne les resultats de recherche
23
24 " -- Beep
25 set visualbell       " Empeche Vim de beeper
26 set noerrorbells    " Empeche Vim de beeper
27
28 " Active le comportement 'habituel' de la touche retour en arriere
29 set backspace=indent,eol,start
30
31 " Active la coloration syntaxique
32 syntax enable
33
34 " Utilise la version sombre de Solarized
35 set background=dark
36 colorscheme solarize

```

FIGURE 8: Le th  me Solarized sombre.



```

1 * VIM Configuration - Vincent Jousse
2 * Annule la compatibilite avec l'ancetre Vi : totalement indispensable
3 set nocompatible
4
5 " -- Affichage
6 set title           " Mets a jour le titre de votre fenetre ou de
7                     " votre terminal
8 set number          " Affiche le numero des lignes
9 set ruler            " Affiche la position actuelle du curseur
10 set wrap            " Affiche les lignes trop longues sur plusieurs
11                    " lignes
12
13 set scrolloff=3      " Affiche un minimum de 3 lignes autour du curseur
14                     " (pour le scroll)
15
16 " -- Recherche
17 set ignorecase       " Ignore la casse lors d'une recherche
18 set smartcase        " Si une recherche contient une majuscule,
19                     " re-active la sensibilit   a la casse
20 set incsearch        " Surligne les resultats de recherche pendant la
21                     " saisie
22 set hlsearch         " Surligne les resultats de recherche
23
24 " -- Beep
25 set visualbell       " Empeche Vim de beeper
26 set noerrorbells    " Empeche Vim de beeper
27
28 " Active le comportement 'habituel' de la touche retour en arriere
29 set backspace=indent,eol,start
30
31 " Active la coloration syntaxique
32 syntax enable
33
34 " Utilise la version sombre de Solarized
35 set background=light
36 colorscheme solarize

```

FIGURE 9: Le th  me solarized clair.

d'entre vous sont des esthètes aguerris.

Si vous êtes sous Mac Os X je vous conseille la police Monaco qui est assez conviviale. Rajoutez les lignes suivantes à votre `.vimrc` pour l'utiliser :

```
set guifont=Monaco:h13
set antialias
```

Listing 5: Utilisation de la police Monaco sous Mac Os X.

Vous pouvez bien sûr changer le h13 par h12 si vous voulez une police plus petite (ou par h14 si vous en voulez une plus grande).

Sinon sous Linux j'utilise la police DejaVu Sans Mono que je trouve plutôt sympathique :

```
set guifont=DejaVu\ Sans\ Mono\ 10
set antialias
```

Listing 6: Utilisation de la police DejaVuSansMono sous Linux.

Vous pouvez là aussi bien sûr changer la taille de la police si vous le souhaitez. Pour avoir la liste des polices disponibles tapez en mode normal `:set guifont:*`.

Vous trouverez une version complète du fichier de configuration pour ce chapitre en ligne <https://github.com/vjousse/vim-for-humans/blob/master/syntaxhighlight/vimrc> ou avec les fichiers mis à disposition avec ce livre. Je ne m'attarderai pas plus sur les polices, c'est assez dépendant de votre système d'exploitation, et un peu moins de Vim dans le coup.

L'explorateur de fichiers : notre premier plugin

Nous y voilà, nous avons un Vim à peu près utilisable avec de jolies couleurs. Maintenant, il faudrait être capable d'ouvrir des fichiers autrement qu'en faisant Fichier (File) -> Ouvrir (Open). Ça va être une bonne occasion pour installer notre premier plugin (ce n'est pas comme si nous avions d'autres choix de toute façon). Nous allons procéder ici en deux étapes, tout d'abord installer un gestionnaire de plugins pour éviter que ça devienne trop le bazar dans vos plugins, puis installer ensuite le plugin qu'il nous faut pour explorer un répertoire de fichiers.

Gestionnaire de plugins : Pathogen

Pathogen²⁴ est typique du plugin que vous découvrirez après avoir commencé à configurer votre Vim et qui génère ce type de réaction :

24. <https://github.com/tpope/vim-pathogen/>

"Ah si j'avais su j'aurais directement commencé avec". Ça tombe bien, c'est ce que nous allons faire.

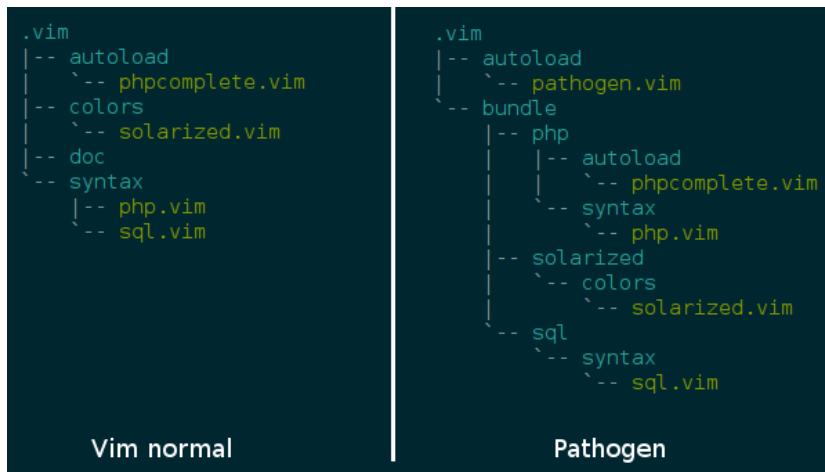
Tout d'abord, petite explication de comment on installe et configure des plugins dans *Vim*. Ils s'installent en copiant les fichiers adéquats (la plupart du temps avec une extension en **.vim*) dans des sous-répertoires de votre répertoire de configuration *.vim*. On a déjà d'ailleurs commencé à y créer un sous-répertoire *colors* qui contient notre "plugin" de coloration *Solarized*.

Le problème avec cette approche c'est que les différents plugins ne sont pas isolés (vous allez devoir copier leurs fichiers dans les différents sous-répertoires) et que vous allez donc vous retrouver avec des fichiers un peu partout sans savoir à qui ils appartiennent. Autant vous dire qu'une fois que vous voulez désinstaller ou mettre à jour un plugin, c'est vite l'enfer pour savoir quels sont ses fichiers.

C'est là que *pathogen* arrive à la rescousse, il va vous permettre d'installer chaque plugin dans un sous-répertoire rien que pour lui. La figure 10 vous donne un exemple de répertoire *.vim* avant et après l'utilisation de *pathogen*. Certes la version avec *pathogen* contient plus de sous-répertoires, mais croyez-moi sur parole, ce rangement va vous éviter bien des ennuis par la suite ²⁵.

25. Et vous pourrez au passage très facilement utiliser *git* pour gérer chacun de vos plugins comme des sous-modules, ce qu'il est impossible de réaliser sinon.

FIGURE 10: *.vim* avant et après *Pathogen*.



Commençons par installer *pathogen*. Créez un répertoire nommé *autoload* dans votre répertoire *.vim* et copiez y *pathogen.vim* que vous pouvez télécharger ici : <https://raw.githubusercontent.com/tpope/vim-pathogen/master/autoload/pathogen.vim> (ou qui vous a été fourni avec ce PDF). Pour les utilisateurs Unix, le listing 7 explique comment l'installer ²⁶.

26. Si vous n'avez pas *curl* vous pouvez aussi utiliser `wget -O -`

```
# Creation du repertoire autoload
mkdir -p ~/.vim/autoload

# Telechargement et installation de pathogen
curl -so ~/.vim/autoload/pathogen.vim \
  https://raw.githubusercontent.com/tpope/vim-pathogen/master/autoload/pathogen.vim
```

Listing 7: Installation de pathogen.

Nous installerons ensuite nos plug-ins directement dans le répertoire `.vim/bundle` que vous allez vous empresser de créer, cf. le listing 8.

```
# Creation du repertoire bundle
mkdir -p ~/.vim/bundle
```

Listing 8: Création du répertoire d'installation des plug-ins.

Il ne vous reste plus qu'à activer pathogen dans votre `.vimrc` et le tour est joué. Nous placerons le code listé dans 9 au début du fichier `.vimrc`, directement après la ligne `set nocompatible`.

```
" Activation de pathogen
call pathogen#infect()
```

Listing 9: Activation du plugin pathogen.

Puisque charité bien ordonnée commence par soi-même, nous allons ranger notre petit plugin solarized en utilisant pathogen. Il nous suffit de créer un répertoire solarized dans notre répertoire `bundle` fraîchement créé²⁷. Nous déplaçons ensuite le répertoire `colors` dans le répertoire solarized (cf. le listing 10).

```
# Creation du repertoire pour solarized
mkdir ~/.vim/bundle/solarized
# Et hop un peu de rangement
mv ~/.vim/colors ~/.vim/bundle/solarized
```

Listing 10: Utilisation de solarized via pathogen.

Voilà notre *Vim* est presque prêt pour le grand bain. Il vous reste une petite étape à franchir : disposer d'un moyen pratique pour explorer les fichiers d'un projet, c'est ici que *The NERD Tree* entre en lice.

Explorateur de fichiers : The NERD Tree

The NERD Tree est un plugin permettant d'afficher visuellement une arborescence de fichiers directement dans la partie gauche

27. Vous pouvez l'appeler comme vous le souhaitez, tout sous-répertoire du répertoire `bundle` sera considéré comme un répertoire de plugin.

(par défaut) de votre *Vim*, à la *TextMate*, *Sublime Text* ou encore *Eclipse/Netbeans*. Ce plugin n'est pas essentiel si vous souhaitez tout contrôler au clavier (je ne l'utilise plus moi-même), mais est assez pratique lorsque l'on débute avec *Vim*.

L'alternative que nous verrons plus tard au chapitre **TODO!** est d'utiliser les plugin *Ctrl-p* ou *Command-t* pour trouver des fichiers et les plugins *LustyExplorer* et *LustyJuggler* pour naviguer entre les fichiers. En effet, devoir visualiser l'arborescence pour trouver un fichier est toujours plus lent que de trouver le fichier à partir de son nom par exemple. The NERD Tree vous permettra donc d'obtenir un *Vim* se comportant comme un éditeur classique avec un explorateur de fichiers sur lequel vous pourrez cliquer.

Nous allons tout d'abord préparer *pathogen* pour installer les différents fichiers de *The NERD Tree*.

```
# Creation du repertoire pour The NERD Tree
mkdir ~/.vim/bundle/nerdtree
```

Listing 11: Création du répertoire pour The NERD Tree.

Téléchargez ensuite le dernier *.zip* disponible sur la page du plugin http://www.vim.org/scripts/script.php?script_id=1658. À l'heure où j'écris ces lignes la dernière version disponible est la version 4.2.0 disponible en téléchargement à cette adresse²⁸ : http://www.vim.org/scripts/download_script.php?src_id=17123.

Ouvrez le fichier *zip* et placez son contenu dans le répertoire `~/.vim/bundle/nerdtree` que nous venons de créer. Vous devriez avoir une arborescence ressemblant à celle ci-dessous pour votre répertoire *nerdtree* :

```
nerdtree
|-- doc
|   '-- NERD_tree.txt
|-- nerdtree_plugin
|   |-- exec_menuitem.vim
|   '-- fs_menu.vim
|-- plugin
|   '-- NERD_tree.vim
'-- syntax
    '-- nerdtree.vim
```

Il va ensuite falloir activer le plugin. Vous pouvez le faire manuellement en tapant `:NERDTree` en mode normal. Si vous préférez activer *The NERD Tree* à chaque fois que vous ouvrez votre *Vim*, ajoutez ces lignes dans votre *.vimrc* :

28. C'est la version que vous trouverez dans les fichiers mis à disposition avec ce PDF

```
" Activation de NERDTree au lancement de vim
autocmd vimenter * NERDTree
```

Listing 12: Activation de NERDTree au lancement de *Vim*.

Rien de particulier ensuite, *The NERD Tree* vous affiche l'arborescence du répertoire où vous avez lancé *Vim*, comme vous le montre la figure 11. Vous pouvez utiliser la souris et/ou le clavier pour vous déplacer.

Vous pouvez aussi effectuer des commandes (créer, copier des fichiers) en appuyant sur la touche **m** lorsque vous êtes dans *The NERD Tree*. Pour passer de la fenêtre de *NERD Tree* à la fenêtre d'édition de votre fichier au clavier, appuyez sur **Ctrl + w + w**²⁹. Ce raccourci clavier sera d'ailleurs toujours valable pour naviguer entre vos différentes fenêtres *Vim* (il n'est pas spécifique à *The NERD Tree*).

29. La touche *Control* (Ctrl) et tout en la laissant appuyée, deux fois **w**.

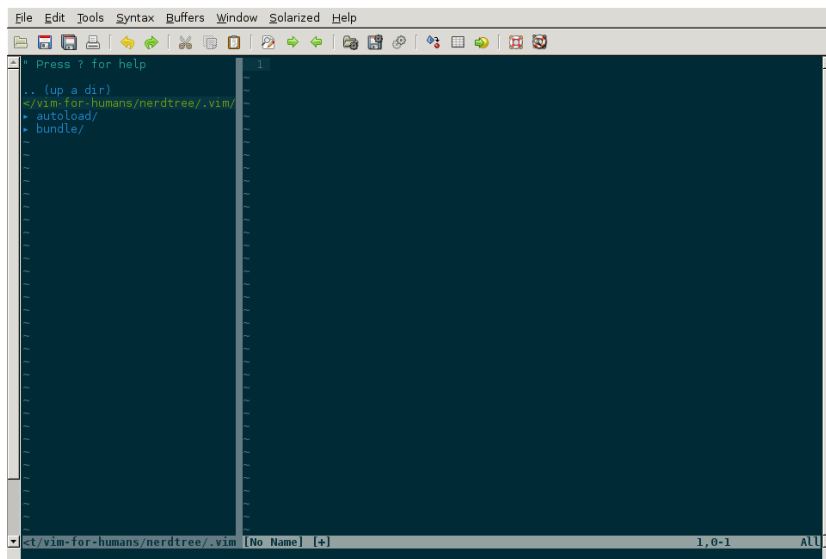


FIGURE 11: *.vim* avec *The NERD Tree* d'activé.

Nous voilà fin prêts

Nous venons de couvrir ce qui pour moi manque cruellement à *Vim* : une configuration par défaut acceptable. Je ne dis pas que c'est la panacée pour l'instant, mais ça devrait vous permettre d'avoir un *Vim* utilisable comme n'importe quel autre éditeur de texte dont vous ne connaissez pas encore toutes les possibilités. Vous pouvez pour l'instant utiliser la souris pour faire un peu tout ce que vous voulez, heureusement ça ne va pas durer.

Nous allons maintenant aborder ce qui fait que *Vim* est unique et

tant prisé : sa gestion des modes et des commandes pour manipuler le texte. La balle est dans votre camp maintenant : ou vous êtes prêt à changer vos habitudes et à passer à un autre niveau d'efficacité, ou alors n'utiliser *Vim* que comme un bloc-notes amélioré vous convient³⁰. C'est vous qui voyez !

30. Dans ce cas là, vous pouvez vous arrêter là.

L'outil de manipulation de texte rêvé

Alors oui, pour ceux qui se demandent, je fais des rêves bizarres, mais bon chacun a ses petites tares cachées. Et rêver d'un outil qui améliore ma vie quotidienne en tant que codeur (ou écrivain, ou formateur, ou ...) n'est pas si étrange que ça.

Ce qui fait et fera encore le succès de *Vim* est sa capacité à **faciliter les manipulations de texte**. Certes il va vous proposer des fonctionnalités propres à chaque tâche que vous effectuerez ³¹ comme la validation syntaxique de code, la correction orthographique, ... Mais à la fin, c'est toujours à écrire/corriger/manipuler/se déplacer dans du texte que vous passerez la majeure partie de votre temps.

C'est là que l'approche de *Vim* est différente d'IDE comme Eclipse / Netbeans / PhpStorm et consorts. Là où ces IDE vont mettre l'accent sur les particularités de votre langage de programmation tout en vous fournissant des capacités de manipulation de texte basiques, *Vim* adopte l'approche opposée : vous serez **très efficace** à manipuler/écrire du texte quel que soit le texte et vous pourrez enrichir *Vim* avec des fonctionnalités propres à votre langage de programmation via des plugins.

Nous allons donc voir dans ce chapitre comment utiliser *Vim* à bon escient (vous allez commencer à oublier votre souris) et quelle est la logique derrière tous ces enchaînements de commandes qui paraissent barbares au non-initié. Vous devriez pouvoir, à la fin de ce chapitre, **vous passer de votre souris** pour éditer/manipuler le contenu d'un fichier ³².

Se déplacer par l'exemple : Essayer de copier / coller

Nous avons déjà vu dans la section «[Préambule indispensable : le mode insertion](#)» comment passer du mode insertion (pour saisir du texte) au mode normal (*a priori* pour l'instant, vous ne savez pas trop à quoi sert ce mode). En appuyant sur la touche **i** votre curseur passe en mode insertion (lorsque vous êtes en mode normal) et en appuyant sur la touche **Esc** (**É**chap) il repasse dans le mode normal. Bon bah on est bien Tintin. Et maintenant ?

31. Souvent par l'intermédiaire de plugins.

32. En tout cas, vous devriez vous forcer à le faire en apprenant *Vim*, ce n'est pas si dur que ça, et c'est ce qui fait la différence entre *Vim* et les autres : le tout clavier.

Préambule

Nous allons apprendre notre première manipulation de texte : le copier / coller. J'en vois certains d'entre vous se dire que ça ne sert à rien, car vous savez déjà le faire. Vous passez en mode insertion, vous prenez votre souris (ou vous vous déplacez avec les flèches directionnelles tout en appuyant sur la touche **Shift**) pour sélectionner du texte et vous allez dans le menu Édition puis Copier. Et ensuite menu Édition puis Coller. Bah tiens, essayez pour voir.

Si vous avez suivi la section «[Les modes : d'où Vim tire sa puissance](#)» traitant de la position idéale pour vos mains, vous savez que vous avez fait une ou plusieurs choses que vous devriez vous interdire :

- Vous avez utilisé votre souris
- Vous avez déplacé grandement votre main droite de sa position de repos, pour aller atteindre les flèches directionnelles qui sont très mal placées sur un clavier

Alors certes ce n'est pas grave en soi, mais c'est **inefficace** (se servir de la souris ou déplacer votre main droite vers les touches directionnelles est très lent) et **nuisible** pour vos petites mains. Ceci est votre dernière chance : si vous n'êtes pas prêt à vous forcer à ne pas le faire, **Vim n'est pas fait pour vous**. Vim est parfait pour ne pas utiliser la souris et pour ne pas bouger vos mains (ou presque). Ne pas se forcer à le faire, c'est ne pas tirer partie de tout le potentiel de Vim, et à un moment ou un autre, **vous le quitterez pour un éditeur** qui aura été pensé pour être utilisé à la souris. Alors, on continue ?

Se passer de la souris

Si vous lisez ces lignes c'est que vous avez répondu « oui », allons-y gaiement alors ! Nous allons tout d'abord commencer par nous passer de la souris. La prochaine étape sera de se passer des touches directionnelles, mais chaque chose en son temps.

POUR RÉALISER UN COPIER/COLLER avec Vim tout se passe en mode « normal ». Pour savoir dans quel mode vous vous trouvez, vous avez juste à regarder en bas à gauche de votre Vim. La figure 12 vous montre Vim en mode « insertion » par exemple. Lorsque rien n'est marqué en bas à gauche, c'est que vous êtes en mode normal. Pour sortir d'un mode afin de retourner au mode normal, il suffit d'appuyer sur la touche **Esc** (**Échap**)³³.

Admettons donc que vous êtes en mode « normal » et que vous avez un peu de texte de saisi dans votre Vim. Par exemple, cette chouette citation de Mark Twain : « Ils ne savaient que c'était impossible, alors ils l'ont fait. ». Votre Vim devrait ressembler à celui de

33. Si vous vous demandez pourquoi je vous dis d'arrêter d'utiliser la souris et/ou les touches directionnelles, mais que je ne dis rien sur le fait qu'il faille se torturer la main pour atteindre la touche **Esc** (**Échap**), c'est que vous êtes sur la bonne voie. Je vous explique le comment du pourquoi dans «[Se passer de la touche Échap](#)».

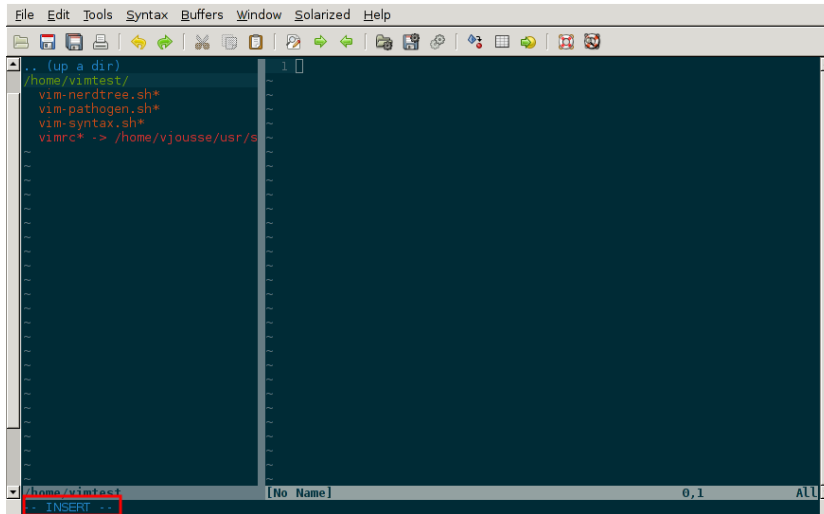
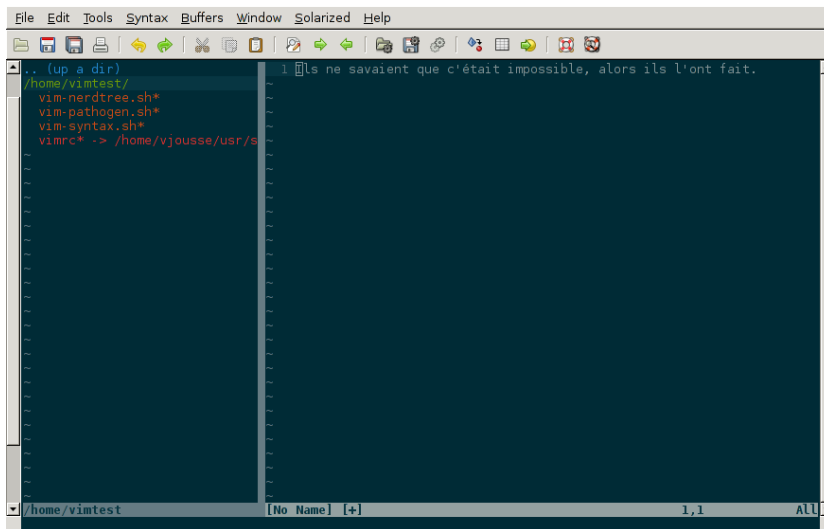


FIGURE 12: Vim en mode insertion.

la figure 13³⁴.



34. Notez l'absence d'affichage d'un quelconque mode en bas à gauche.

FIGURE 13: Vim prêt pour le copier/coller.

La façon la plus naturelle³⁵ de copier/coller le mot « impossible » va être de se déplacer sur la première lettre du mot avec les touches directionnelles, d'appuyer sur la touche **v** (pour passer en mode « visuel »), de se déplacer sur la dernière lettre (vous devriez avoir le mot sélectionné, en surbrillance) puis d'appuyer sur la touche **y**³⁶. Vous avez copié votre premier mot.

Déplacez vous ensuite à la fin de la phrase (toujours en mode « normal ») puis appuyez sur la touche **p**³⁷. Le mot devrait avoir été collé à la fin, et vous devriez avoir le même rendu que la figure 14.

35. Mais pas la plus efficace, nous verrons cela un peu plus loin.

36. La touche **y** étant utilisée comme raccourci du mot *yank* en anglais.

37. Raccourci du mot *paste* cette fois ci.

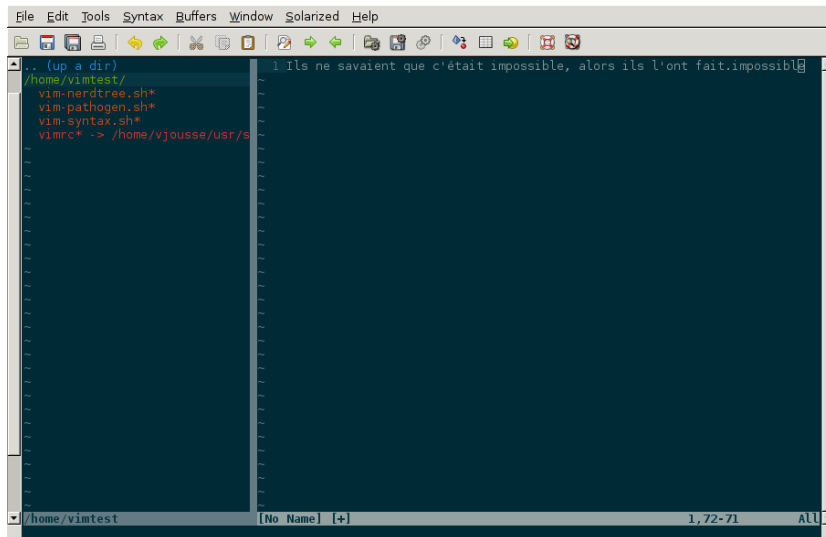


FIGURE 14: Vim après le copier/coller.

On se rend donc compte ici que *Vim* se sert de l’astuce des modes (et notamment du mode « normal » pour les déplacements) afin de ne pas avoir à se servir de la souris. À partir du moment où vous aurez pris l’habitude de passer rapidement d’un mode à l’autre (et pour cela se passer de la touche **Esc** (**Échap**) va devenir indispensable), utiliser la souris vous apparaîtra comme une perte de temps, mais pour cela il va falloir pratiquer un peu bien sûr.

Se passer des touches directionnelles

Nous y voilà. Encore plus que de se priver de la souris, se priver des touches directionnelles est la chose à faire si l’on veut utiliser *Vim*, pour de vrai. *Vim* va vous permettre de faire tout plus rapidement et plus intuitivement à la seule condition de le faire sans les touches directionnelles. Cela va vous permettre comme je l’ai déjà dit de ne pas bouger votre main certes, mais ça va aussi vous forcer à passer en mode « normal » pour réaliser vos déplacements et vos mouvements de texte. Il n’y a qu’à ce moment là³⁸ que vous commencerez à vraiment tirer parti de *Vim*.

Pour cette section, je vais vous expliquer comment vous déplacer sans utiliser les touches directionnelles. Puis, une fois que vous aurez une vague idée de comment faire, je vous donnerai le code à mettre dans votre *.vimrc* pour désactiver les touches directionnelles complètement. Car oui, il n’y a que comme ça que vous y arriverez (en tout cas il n’y a que comme ça que j’y suis arrivé).

38. Un peu douloureux au début il est vrai.

Se déplacer sans les touches directionnelles

En mode normal, 4 touches vont vous permettre de déplacer le curseur d'un caractère :

- la touche **h** pour aller **à gauche**
- la touche **j** pour aller **en bas**
- la touche **k** pour aller **en haut**
- la touche **l** pour aller **à droite**

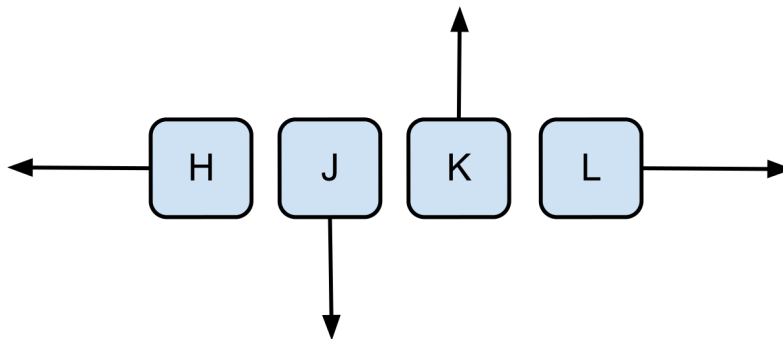


FIGURE 15: Les « touches directionnelles » de Vim en mode normal.

Vous pouvez remarquer que ces touches sont placées sur la rangée de repos de manière à déplacer vos doigts le moins possible. En essayant de placer vos doigts pour atteindre ces lettres vous devriez vous rendre compte que l'index a deux déplacements (gauche et bas) alors que l'auriculaire n'en a pas. Vous verrez qu'on s'y fait assez rapidement et que l'index étant plus fort que l'auriculaire, ça tombe plutôt bien³⁹.

À noter qu'à force, on se sert de moins en moins des déplacements gauche/droite d'un caractère. On va leur préférer les déplacements de mot en mot, de paragraphe en paragraphe ou les déplacements grâce à des recherches. Quelques exemples de déplacements "rapides" que j'utilise :

Touche	Déplacement
e	à la fin du mot courant
b	au début du mot courant
w	au début du mot suivant
^	au premier caractère non blanc de la ligne
\$	à la fin de la ligne
0	au début de la ligne

Vous avez ici le minimum pour vous déplacer en mode normal. Il existe aussi des commandes vous permettant de vous déplacer puis de rentrer en mode insertion directement, elles sont très pratiques car

39. Vous trouverez le clavier sur lequel Vi a été conçu dans la section «[Se passer de la touche Échap](#)», vous comprendrez ainsi le pourquoi du comment.

elles vont vous permettre d'économiser quelques touches. En voici quelques unes que j'utilise à peu près tout le temps :

Touche	Action
i	se place en mode insertion avant l'emplacement du curseur
a	se place en mode insertion après l'emplacement du curseur
I	se place en mode insertion au début de la ligne
A	se place en mode insertion à la fin de la ligne
o	insère une nouvelle ligne en dessous de la ligne courante
O	insère une nouvelle ligne au dessus de la ligne courante
r	remplace les caractères sous le curseur

Arrêtons-nous un peu là dessus. Au risque d'insister lourdement, mais la clé de l'utilisation de *Vim* vient de ce que nous venons de voir dans ce chapitre, ni plus, ni moins. Il y a une chose que vous avez à vous forcer à faire, c'est **d'utiliser les touches hjkl** pour les déplacements. Si vous y arrivez, tout le reste vous l'apprendrez au fur et à mesure.

Vous trouverez des sites entiers vous détaillant les différentes commandes possibles, les différentes combinaisons, j'en passe et des meilleures. Vous les apprendrez puis les oublierez (ou pas, en fonction de si elles vous sont vraiment utiles). Si vous avez un seul effort à faire c'est celui de se passer des touches directionnelles et donc de vous forcer à utiliser le mode normal. Le reste tombera sous le sens.

Voici l'ultime configuration qu'il vous faudra mettre dans votre *.vimrc* pour atteindre le Saint Graal : désactiver les touches directionnelles.

```
" Désactiver les touches directionnelles
map <up> <nop>
map <down> <nop>
map <left> <nop>
map <right> <nop>
imap <up> <nop>
imap <down> <nop>
imap <left> <nop>
imap <right> <nop>
```

Listing 13: Désactiver les touches directionnelles.

Nous y voilà. Croyez-moi, vous allez souffrir un peu au début. Pour moi, ça n'a pas dû durer plus de 2 jours. Ensuite vous aurez oublié. Si vous n'êtes pas prêt à galérer un peu pendant deux jours pour améliorer votre efficacité à vie, que faites-vous ici !

Je ne vous donnerai pas d'autres détails sur toutes les touches possibles pour vous déplacer, d'autres ressources le font déjà bien mieux que moi. Je vais en revanche vous apprendre dans [Combiner des touches/déplacements](#) comment les utiliser à bon escient.

On peut notamment citer le livre gratuit "A byte of Vim" traduit en français et disponible à l'adresse suivante : http://swaroopch.com/notes/Vim_fr/.

Ou encore l'infographie de la figure 16⁴⁰ qui donne un aperçu des différents mouvements pour chacune des touches d'un clavier français.

40. Téléchargeable sur <http://www.nathael.org/>

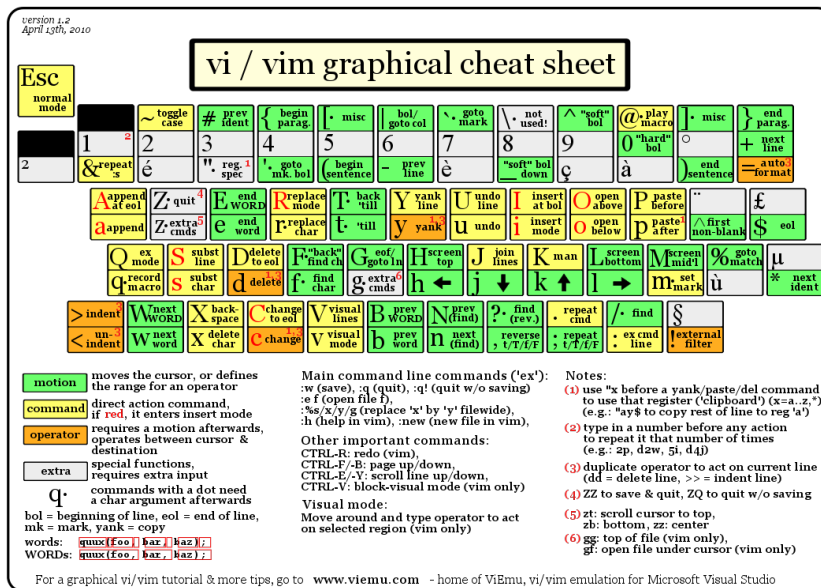


FIGURE 16: Les touches Vim.

N'oubliez pas que le but ici est de gagner en rapidité en ne bougeant quasi plus ses mains de la rangée de repos, et en utilisant le plus possible le « mode normal ». Au boulot!

Se passer de la touche Échap

Utiliser la touche **Esc** (Échap) pour sortir du mode « insertion » semble être une hérésie tellement elle est difficilement accessible. Il faut déplacer entièrement la main gauche pour y accéder ou alors se torturer le petit doigt.

Pour comprendre pourquoi la touche **Esc** (Échap) est utilisée par défaut, il faut faire un bon de quelques années en arrière, pour se retrouver en face du clavier sur lequel Vi a été développé. Vous pouvez voir sur la photo 17 que la touche **Esc** (Échap) était très facilement accessible. Vous pouvez aussi noter l'emplacement des touches direc-

tionnelles. Malheureusement depuis, cela a bien changé.

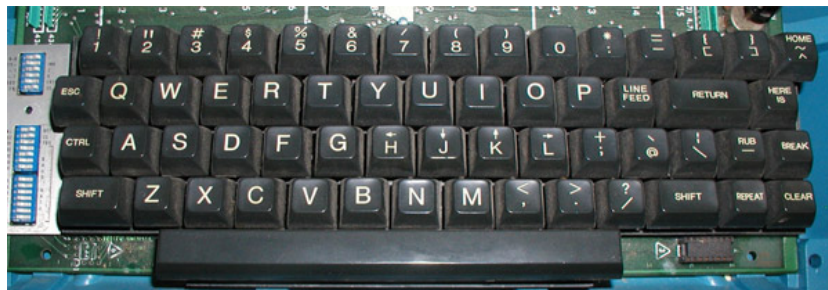


FIGURE 17: Le clavier sur lequel *Vi* a été réalisé.

L'étape ultime (après avoir réussi à se passer des touches directionnelles) est donc de rapprocher la touche **Esc** (**Échap**) de vos petits doigts. Il y a plusieurs solutions pour cela, mais celle que je vous recommande si vous avez un clavier avec une disposition française est la suivante (dans votre *.vimrc*) :

```
" Les ; sons rarement utilise l'un a la suite de l'autre
:imap ;; <Esc>
```

Listing 14: Taper deux fois sur **;** pour quitter le mode normal.

Lorsque vous êtes en mode insertion, il vous suffit d'appuyer deux fois sur la touche **;** pour retourner au mode normal. la touche **;** ne vous demande pas de bouger votre main de la rangée de repos et on l'utilise rarement deux fois de suite (et si c'est le cas, il suffit d'attendre un peu avant de taper le deuxième **;**), c'est donc le parfait candidat.

Voici d'autres solutions possibles (cf http://vim.wikia.com/wiki/Avoid_the_escape_key) :


```

:imap jj <Esc>

:imap jk <Esc>

:imap ii <Esc>

:imap ' <Esc>

" Shift-Espace (peut ne pas marcher sur votre systeme).
:imap <S-Space> <Esc>

" Sous Linux avec gvim Vim en console, vous pouvez utiliser Alt-Space.
:imap <M-Space> <Esc>

```

Listing 15: D'autres combinaisons de touches possibles pour quitter le mode normal.

Combiner des touches/déplacements

Maintenant que nous savons nous déplacer en mode normal, il est temps de voir comment réaliser d'autres opérations. Nous avons déjà vu le copier/coller au chapitre [Se déplacer par l'exemple : Essayer de copier / coller](#), nous allons maintenant voir comment supprimer/éditer plus facilement.

Dans [Se passer des touches directionnelles](#) nous avons vu qu'il suffisait d'utiliser la touche **w** pour se déplacer au début du mot suivant. Nous allons essayer de combiner cela avec quelques nouvelles touches du mode normal :

- la touche **d** est utilisée pour « supprimer »
- la touche **c** est utilisée pour « supprimer et passer en mode insertion »

À noter que ce qui est supprimé est placé dans le presse-papier en même temps (le « supprimer » se comporte par défaut comme un « couper »).

La particularité de ces touches, c'est qu'elles attendent ensuite un « ordre de déplacement » pour savoir quoi supprimer. Il va donc falloir les combiner avec les déplacements que nous avons déjà vus dans [Se passer des touches directionnelles](#).

Cela donnera par exemple :

Action	Résultat
la touche d puis la touche w	supprime les caractères jusqu'au prochain mot
la touche c puis la touche w	supprime les caractères jusqu'au prochain mot et passera en mode insertion
la touche d puis la touche \$	supprime tout jusqu'à la fin de la ligne
la touche d puis la touche ^	supprime tout jusqu'au début de la ligne

Vous pouvez aussi utiliser cela pour copier :

Action	Résultat
la touche y puis la touche w	copie les caractères jusqu'au prochain mot
la touche y puis la touche \$	copie tout jusqu'à la fin de la ligne
la touche y puis la touche ^	copie tout jusqu'au début de la ligne

Il ne vous restera qu'à appuyer sur la touche **p** pour coller ce que vous voulez où vous voulez. Par défaut la touche **p** colle le texte après la position courante du curseur. Si vous voulez coller avant la position du curseur, utilisez la touche **P**.

Il arrive de temps en temps de vouloir aussi supprimer du texte (non sans blague !), voici quelques commandes utiles pour cela :

Action	Résultat
dd	efface la ligne courante et la place dans le presse-papier
x	efface le caractère sous le curseur
X	efface le caractère avant le curseur

La plupart des mouvements peuvent être préfixés par un nombre multiplicateur. Voici quelques exemples :

Action	Résultat
2dd	efface deux lignes
3x	efface 3 caractères vers l'avant du curseur
3X	efface 3 caractères vers l'arrière du curseur
2yy	copie 2 lignes dans le presse-papier
5j	se déplace de 5 lignes vers le bas

Rechercher / Se déplacer rapidement

Maintenant que nous connaissons les commandes de base pour éditer du texte avec *Vim*, voyons voir comment nous déplacer plus rapidement dans notre document. Nous avons déjà évoqué les touches **w**, **b**, **^** et **\$** qui nous permettent respectivement de se déplacer à la fin d'un mot, au début d'un mot, au début d'une ligne et la fin d'une ligne. Tout d'abord, voyons voir comment « scroller » sans la souris. À noter que toutes ces commandes se font en mode « normal ».

Sauts de page

Pour faire défiler les pages, il faut utiliser :

- **Ctrl (Control) + f** pour passer à la page suivante (**f** pour forward)
- **Ctrl (Control) + b** pour passer à la page précédente (**b** pour backward)

Ces raccourcis vont vous permettre d'avancer rapidement dans votre document.

Vous pouvez aussi :

- Vous rendre au début du fichier en tapant **gg**
- Vous rendre à la fin du fichier en tapant **GG**
- Vous rendre à la ligne 23 en tapant **:23**

Les marqueurs

Lorsque je me déplace dans un fichier, j'aime bien pouvoir revenir à certains endroits. Par exemple lorsque je me rends au début du fichier alors que j'étais en train de travailler au milieu de celui-ci, j'aime bien pouvoir revenir directement où je travaillais. Heureusement, *Vim* a tout prévu pour cela grâce à l'utilisation de **marqueurs**. Les marqueurs sont tout simplement des « marque-pages » qui permettent à votre curseur de se retrouver à la position où vous aviez mis votre marqueur.

Un marqueur se pose en tapant **ma**. Pour déplacer votre curseur à la position du marqueur tapez **'a**. Vous pouvez placer plusieurs marqueurs en changeant **a** par n'importe quelle lettre de l'alphabet (on appelle cela des registres en langage *Vim*). Pour placer un autre marqueur vous pouvez par exemple utiliser la lettre **d**. Grâce à **md** vous placerez le marqueur et à **'d** vous y rendrez.

La recherche

En mode normal, vous pouvez lancer une recherche en utilisant la touche **/** suivi du texte que vous souhaitez rechercher puis de la touche **Entrée**. Grâce à notre configuration de *Vim* vous devriez voir vos occurrences de recherche surlignées en même temps que vous tapez. Par défaut la recherche n'est pas sensible à la casse (pas de différence entre minuscules/majuscules). En revanche, dès que vous taperez une majuscule, la recherche deviendra sensible à la casse. Vous pouvez vous déplacer à la prochaine occurrence de la recherche grâce à la touche **n**. Pour vous déplacer à la précédente utilisez la touche **N**.

Pour rappel, voici les lignes de votre fichier de configuration qui permettent de faire cela :

```
" -- Recherche
set ignorecase      " Ignore la casse lors d'une recherche
set smartcase       " Si une recherche contient une majuscule,
                    " re-active la sensibilité à la casse
set incsearch       " Surligne les résultats de recherche pendant la
                    " saisie
set hlsearch        " Surligne les résultats de recherche
```

Listing 16: Configuration de la recherche avec *Vim*.


Attention par défaut, la recherche utilise les expressions régulières POSIX. Si vous souhaitez rechercher des caractères habituellement utilisés dans les expressions régulières (comme [] ^ \$ /) n'oubliez pas de les préfixer par \.

Vous pouvez aussi rechercher directement le mot qui est placé sous votre curseur grâce à la touche *. Utiliser la touche * fera une recherche vers l'avant. Pour faire une recherche vers l'arrière, utilisez la touche #.

Le mode visuel

Je vous en ai déjà parlé lors de l'explication sur le Copier / Coller, mais comme je sais que certains d'entre vous sont tête en l'air, je vous fais un petit rappel ici.

Lorsque vous êtes en mode "normal" appuyez sur la touche **v** pour passer en mode "visuel". Vous pourrez alors sélectionner des caractères ou des lignes entières grâce aux différentes façon de vous déplacer que vous venez d'apprendre. Vous pourrez ensuite copier le texte sélectionné avec la touche **y** puis le coller avec la touche **p**. Pour le couper il vous faudra utiliser la touche **d**.

En mode normal vous pourrez utiliser la touche **V** pour sélectionner lignes par lignes. Et bien sûr, utiliser la touche **Esc** (**Échap**) ou  pour revenir au mode normal.

À vous de jouer

Vous devriez maintenant être capable de n'utiliser que le clavier pour les opérations de manipulation de texte et d'édition. Je n'ai fait que survoler la puissance de *Vim* ici, mais ça devrait être suffisant pour survivre. Je vous ai donné ici le strict nécessaire, mais ce strict nécessaire vous permet déjà de profiter de *Vim* et du plaisir de ne plus utiliser la souris.

À vous maintenant de lire les nombreuses ressources disponibles sur internet vous décrivant tous les mouvements possibles et imaginables. Je ne manquerai d'ailleurs pas de compléter ce guide avec des articles sur le site internet qui lui est dédié <http://www.vimebook.com>.

Voici une liste de ressources qui pourraient vous être utiles, malheureusement les ressources en français sont assez rares :

- A byte of *Vim* en français http://www.swaroopch.com/notes/vim_fr/
- Un petit pense bête sympathique de différents raccourcis clavier <http://www.tuteurs.ens.fr/unix/editeurs/vim.html>
- Un wiki non officiel francophone (un peu fouillis soit dit en passant) : www.vim-fr.org/
- Les vidéos Peepcode en anglais mais vraiment superbement réalisées : <https://peepcode.com/products/smash-into-vim-i> et <https://peepcode.com/products/smash-into-vim-ii>
- Le blog de Derek Wyatt's en anglais <http://www.derekwyatt.org/vim/vim-tutorial-videos/>

Histoire de réveiller l'enfant qui est en vous, je vous conseille vivement d'aller vous amuser avec <http://vim-adventures.com/>. C'est un jeu de rôle en ligne qui a pour but de vous apprendre à manipuler *Vim* ! Je vous ai mis un petit aperçu dans l'image numéro 18.



FIGURE 18: Vim adventures, une façon ludique d'apprendre *Vim*.

Nous allons maintenant passer à la vitesse supérieure : l'utilisation de plugins, ou comment rendre *Vim* incontournable.

Les plugins indispensables

Soyons clair, *Vim* sans ses plugins, c'est comme Milan sans Rémo⁴¹ : ça ne rime à rien. C'est grâce aux plugins que *Vim* va pouvoir pleinement exprimer toute sa puissance et vous élever à un autre niveau de productivité. Vous n'avez pas besoin d'en avoir des mille et des cents, mais quelques uns savamment choisis devraient faire l'affaire.

Qu'on ne se méprenne pas, *Vim* peut bien sûr s'utiliser sans plugins. Il peut d'ailleurs s'avérer utile de savoir faire les manipulations de base sans avoir besoin d'installer de plugin, car c'est souvent le cas sur des serveurs : il n'y a aucun plugin d'installé. Dans ce cas là, savoir ouvrir, sauvegarder sous, passer d'un fichier à l'autre avec les commandes de *Vim* par défaut peut vous sauver la mise. En revanche, dans votre travail quotidien de rédaction ou de code, les plugins sont indispensables pour pleinement tirer partie de *Vim*.

Naviguer sur le disque et entre les fichiers : Lusty Explorer

Nous avons déjà vu NerdTree dans [Explorateur de fichiers : The NERD Tree](#) qui permettait d'avoir un explorateur de projet dans une fenêtre latérale de *Vim*. Le problème de ce plugin est qu'il n'est pas fait pour être utilisé au clavier. Certes vous pouvez utiliser le clavier, mais il ne sera pas aussi efficace que les plugins pensés uniquement pour une utilisation au clavier.

Personnellement, le premier plugin que j'installe partout où j'ai à utiliser *Vim*, c'est *Lusty Explorer*⁴². Ce plugin va vous permettre de naviguer sur votre disque dur pour ouvrir facilement des fichiers en se passant de la souris. Il va aussi permettre de naviguer rapidement entre vos différents fichiers déjà ouverts (vos buffers on jargon *Vim*). Commençons par l'installer.

Rendez-vous sur l'url du script http://www.vim.org/scripts/script.php?script_id=1890 et télécharger la dernière version (c'est actuellement la 4.3)⁴³. Faites ensuite le nécessaire dans votre répertoire `.vim/` pour qu'il ressemble à la structure ci-dessous :

```
.vim
```

41. ©François Corbier - Sans ma barbe
- <http://www.bide-et-musique.com/song/149.html>

42. http://www.vim.org/scripts/script.php?script_id=1890

43. http://www.vim.org/scripts/download_script.php?src_id=17529

```
|-- autoloader
|  '-- pathogen.vim
'-- bundle
|  |-- lusty-explorer
|  |  '-- plugin
|  |  '-- lusty-explorer.vim
```

Si vous avez suivi tout ce que l'on a fait depuis le début votre répertoire *.vim/*, il devrait maintenant ressembler à cela :

```
.vim
|-- autoloader
|  '-- pathogen.vim
'-- bundle
|  |-- lusty-explorer
|  |  '-- plugin
|  |  '-- lusty-explorer.vim
|  |-- nerdtree
|  |  |-- doc
|  |  |  '-- NERD_tree.txt
|  |  |-- nerdtree_plugin
|  |  |  |-- exec_menuitem.vim
|  |  |  '-- fs_menu.vim
|  |  |-- plugin
|  |  |  '-- NERD_tree.vim
|  |  '-- syntax
|  |  '-- nerdtree.vim
'-- solarized
|  '-- colors
|  '-- solarized.vim
```

Reste à voir comment l'utiliser. Si l'on se réfère à la documentation, voilà ce que l'on trouve (traduit en français) :

```
<Leader>lf - Ouvre l'explorateur de fichiers.
<Leader>lr - Ouvre l'explorateur de fichiers à partir du répertoire du fichier courant.
<Leader>lb - Ouvre l'explorateur de buffers.
<Leader>lg - Ouvre la recherche dans les buffers.
```

On voit qu'il est fait mention d'une touche nommée **<Leader>** qu'il faut ensuite faire suivre d'autres touches comme *lf*, *lr*, *lb* et *lg*. Cette touche **<Leader>** est une touche spéciale que l'on définit dans son fichier *.vimrc*. Elle sera énormément utilisée par tous les plugins, beaucoup des commandes de ces derniers commenceront par la touche **<Leader>**. C'est un moyen d'éviter les collisions avec les raccourcis par défaut de *Vim*.

Il faut donc choisir une touche **<Leader>**. Par défaut, *Vim* utilise **** comme touche **<Leader>**. Sur nos claviers francophones c'est une très mauvaise idée d'utiliser cette touche car elle n'est pas pratique du tout. La plupart des utilisateurs de *Vim* la remplace par la touche **,** (la virgule). Elle est directement accessible sous l'index de la main droite ce qui en fait une parfaite candidate. Pour spécifier cela à *Vim* il va falloir rajouter une ligne dans votre fichier *.vimrc*, à savoir :

```
let mapleader = ","
```

Listing 17: Spécifier la touche leader.

Une fois la modification effectuée et prise en compte (en redémarrant *Vim* ou en tapant `:source ~/.vimrc` en mode normal), vous devriez être en mesure de taper **,lr** et d'avoir le même style de résultat que sur la figure 19.

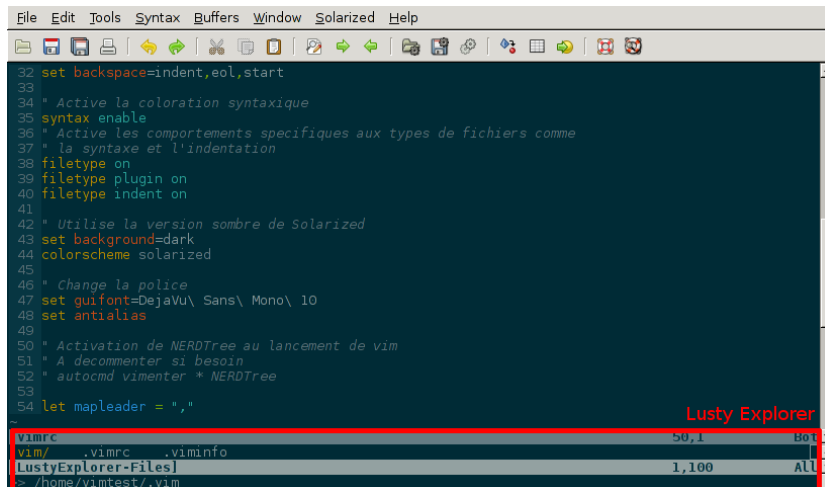


FIGURE 19: Vim avec Lusty Explorer d'activé en bas.

Je vous conseille maintenant de désactiver *The Nerd Tree* (en commentant la ligne au dessus du *mapleader* comme je l'ai fait dans la figure 20), il ne vous servira plus à grand chose, *Lusty Explorer* le remplace à merveille.

Vous pouvez constater sur la figure 19 qu'il y a deux parties à *Lusty Explorer*. La partie basse vous indique le répertoire que vous êtes en train d'explorer et la partie haute liste le contenu de ce répertoire. En surbrillance se trouve l'élément couramment sélectionné. Dans le cas de la figure 19 c'est le répertoire *.vim/* en jaune (la couleur pourra être différente en fonction de votre thème).

Lusty Explorer utilise une fonctionnalité de *Fuzzy matching* qui va vous permettre de ne taper qu'une partie d'un nom de fichier pour

le sélectionner. Dans mon exemple, si, dans la fenêtre de *Lusty*, je saisi `.vimi` il va me sélectionner le fichier `.viminfo` sans que j'ai à lui spécifier le nom entier, je n'aurais ensuite plus qu'à appuyer sur la touche **Entrée** pour ouvrir le fichier dans *Vim*. La figure 20 vous montre l'exemple en question.

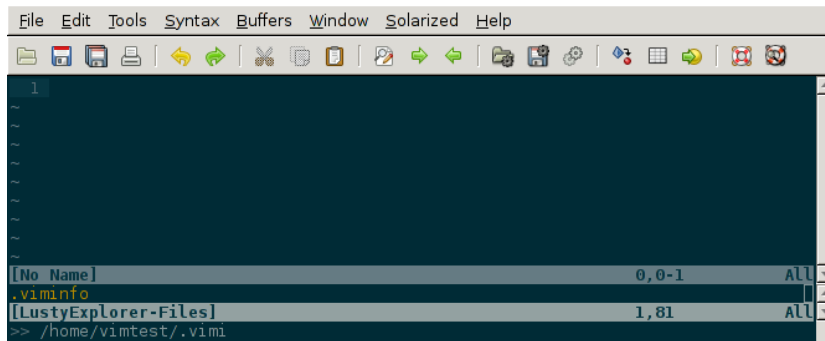


FIGURE 20: Lusty Explorer et le Fuzzy matching.

Lusty Explorer dispose en plus de quelques raccourcis bien pratiques pour utiliser le navigateur de fichiers :

- **Ctrl** (**Control**) + **n** pour sélectionner le fichier/répertoire suivant
- **Ctrl** (**Control**) + **p** pour sélectionner le fichier/répertoire précédent
- **Ctrl** (**Control**) + **w** pour descendre au répertoire parent
- **Ctrl** (**Control**) + **e** crée un nouveau fichier vide (non sauvegardé sur le disque) avec le nom spécifié actuellement dans *Lusty Explorer*. Vous n'aurez plus qu'à utiliser `:w` pour écrire le contenu du fichier sur le disque.

Lusty Explorer s'utilise donc pour deux choses : naviguer sur votre système de fichiers avec `,lr` et `,lf`, et naviguer entre vos fichiers ouverts (buffers) avec `'lb`. Personnellement j'utilise moins la recherche dans les buffers avec `,lg`, à vous de tester et de vous faire votre propre opinion.

Je vous conseille en guise de test d'ouvrir plusieurs fichiers avec `,lr` ou `,lf`. Ensuite, entraînez-vous à naviguer entre ces différents fichiers ouverts en même temps à l'aide de `,lb`. C'est une des combinaisons que j'utilise le plus au quotidien.

Ce plugin est indispensable et ajoute à lui seul énormément de valeur à *Vim* : se passer de la souris pour ouvrir des fichiers. Prenez donc le temps nécessaire pour l'apprendre correctement, c'est un investissement qui vaut le coup.

Recherche dans les fichiers sur le disque : Ack

Lorsque l'on édite un fichier appartenant à un projet plus gros contenant lui même beaucoup de fichiers, il arrive souvent de vouloir rechercher une occurrence d'une chaîne de caractères dans tous les fichiers du projet. Pour ce faire, *Vim* dispose d'un plugin permettant d'utiliser *Ack* pour faire cette recherche.

*Ack*⁴⁴ est un programme écrit en *perl* qui remplace avantageusement le bon vieux *grep* pour effectuer des recherches dans des fichiers. Il a en revanche un désavantage par rapport à *grep* : il est rarement installé par défaut. Nous allons donc commencer par installer *Ack* avant de pouvoir aller plus loin. Cela va bien sûr dépendre de la plateforme sur laquelle vous utilisez *Vim*, vous pourrez trouver différentes instructions en fonction de votre plateforme sur la page du plugin : <http://github.com/mileszs/ack.vim#installation>.

Pour Debian/Ubuntu : `sudo apt-get install ack-grep`. Pour Mac Os X vous allez avoir besoin de Homebrew (<http://mxcl.github.com/homebrew/>) en utilisant `brew install ack`. Pour les utilisateurs de MacPorts ça sera avec la commande `sudo port install p5-app-ack`. Pour Windows installez Strawberry Perl (<http://strawberryperl.com/>) et dans le shell de commandes exécutez `C:\>cpan App : :Ack`. Vous devriez ensuite pouvoir utiliser la commande `ack` dans votre terminal de commandes en lieu et place de `grep`.

Rendez-vous sur la page du plugin *ack*⁴⁵ et téléchargez la dernière version (à l'heure où j'écris ces lignes c'est la version 0.3.1). Décompressez l'archive dans votre répertoire `~/vim/bundle/`, de manière à obtenir une structure de ce type :

```
bundle
|-- ack
|   |-- doc
|   |   '-- ack.txt
|   '-- plugin
|       '-- ack.vim
...
```

Comme d'habitude assurez-vous que vos modifications sont bien prises en compte en redémarrant *Vim* ou en tapant `:source ~/.vimrc` en mode normal.

Il va ensuite falloir ajouter quelques lignes à notre fichier `.vimrc` pour faciliter d'utilisation du plugin :

44. <http://betterthangrep.com/>

45. http://www.vim.org/scripts/script.php?script_id=2572

```
" Parametres par default pour ack
let g:ackprg="ack -H --nocolor --nogroup --column"
" Place un marqueur et cherche
nmap <leader>j mA:Ack<space>
" Place un marqueur et cherche le mot sous le curseur
nmap <leader>ja mA:Ack "<C-r>=expand("<cword>")<cr>"
nmap <leader>JA mA:Ack "<C-r>=expand("<cWORD>")<cr>"
```

Listing 18: Configuration du plugin Ack.

Ack recherchera alors à partir du répertoire où se trouve votre fichier couramment ouvert. Vous pouvez faire quelques tests si vous le souhaitez (en supposant que votre touche <leader> est la touche , (la virgule) :

- `,j` toto : recherchera toto à partir du répertoire du fichier courant,
- `,ja` avec votre curseur sur un mot recherchera ce mot.

Le plugin Ack vous affichera les résultats dans une fenêtre que l'on appelle *Quickfix Window*, cf image 21.

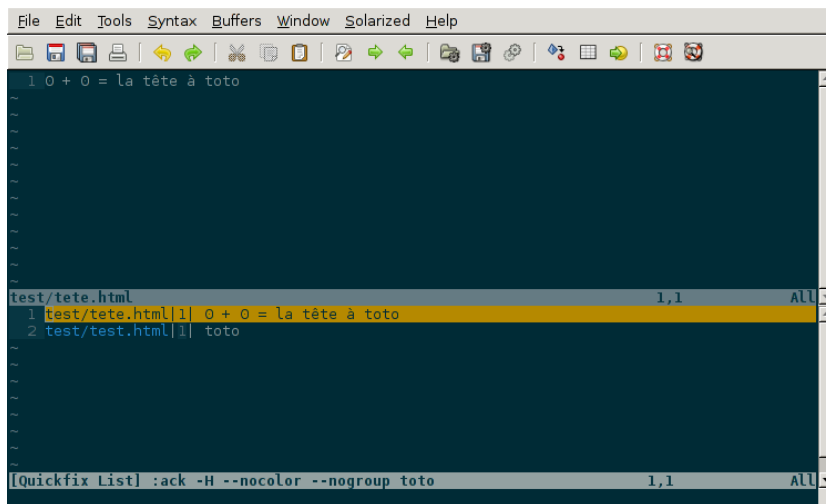


FIGURE 21: Vim avec la fenêtre Quickfix de Ack.

Voici quelques commandes disponibles dans cette fenêtre :

- `o` : ouvrir (idem que <Entrée>)
- `go` : voir un aperçu (ouvre le fichier mais maintient le focus sur les résultats de ack.vim)
- `t` : ouvrir dans un nouvel onglet
- `T` : ouvrir dans un nouvel onglet en arrière plan
- `h` : ouvrir en séparant la fenêtre horizontalement
- `v` : ouvrir en séparant la fenêtre verticalement

- **q** : fermer la fenêtre quickfix

À noter que par défaut Ack ne recherche que dans les fichiers qu'il reconnaît comme pertinents (il ne fera pas de recherche dans les fichiers temporaires, les fichiers des gestionnaires de version, etc.). Si vous souhaitez que Ack recherche dans tous les fichiers indépendamment de leur type, vous devez spécifier l'option `-u` comme ceci dans votre `.vimrc` :

```
" Parametres par default pour ack
let g:ackprg="ack -H -u --nocolor --nogroup --column"
```

Listing 19: Configuration du plugin Ack pour rechercher dans tous les fichiers.

Recherche de fichiers sur le disque

Non ce n'est pas pareil que Ack, relisez bien le titre. Ici nous n'allons pas chercher dans les fichiers, mais nous allons plutôt chercher des fichiers à ouvrir avec *Vim*. Ça peut s'avérer très utile lorsque vous avez à travailler sur des projets où les fichiers sont éparpillés dans un grand nombre de répertoires.

Comme d'habitude nous allons commencer par installer le plugin. Une fois n'est pas coutume, le plugin dispose d'une page dédiée plutôt bien réalisée que vous trouverez ici : <http://kien.github.com/ctrlp.vim/>. Scrollez tout en bas pour télécharger la dernière version en "Direct Downloads". Pour les paresseux, voici un lien direct : <http://github.com/kien/ctrlp.vim/zipball/master>. Décompressez l'archive dans votre répertoire `~/vim/bundle/`, de manière à obtenir une structure de ce type :

```
bundle
|
...
|-- ctrlp
|   |-- autoload
|   |   |-- ctrlp
|   |   |   |-- bookmarkdir.vim
|   |   |   |-- buffertag.vim
|   |   |   |-- changes.vim
|   |   |   |-- dir.vim
|   |   |   |-- line.vim
|   |   |   |-- mixed.vim
|   |   |   |-- mrufiles.vim
```

```

| | | | -- quickfix.vim
| | | | -- rtscript.vim
| | | | -- tag.vim
| | | | -- undo.vim
| | | | -- utils.vim
| | | | -- ctrlp.vim
| | -- doc
| | | | -- ctrlp.txt
| | -- plugin
| | | | -- ctrlp.vim
| | -- readme.md
...

```

Comme d'habitude assurez-vous que vos modifications sont bien prises en compte en redémarrant *Vim* ou en tapant `:source ~/.vimrc` en mode normal.

Nous n'avons plus qu'à ajouter un raccourci dans notre *.vimrc* pour invoquer CtrlP comme le montre le listing 20. Dans mon cas j'ai choisi `,c`, mais vous pouvez choisir ce que vous voulez.

```
let g:ctrlp_map = '<leader>c'
```

Listing 20: Configuration du raccourci pour activer CtrlP.

La figure 22 vous montre CtrlP en action. Il vous suffit de l'invoquer avec `,c` et de taper le début du fichier que vous recherchez. Quand le fichier voulu sera sélectionné en premier, il ne vous restera plus qu'à appuyer sur la touche **Entrée** pour l'ouvrir.

À noter que CtrlP peut aussi être utilisé pour naviguer entre les fichiers ouverts (comme *Lusty*). Mais à l'usage, je le trouve moins pratique que *Lusty*. Vous pouvez aussi vous en servir pour naviguer automatiquement dans votre code en "suivant" vos fonctions grâce aux tags (comme on peut le faire dans *Eclipse*). C'est un trop vaste sujet pour être traité dans ce guide, mais si ça vous intéresse vous pouvez déjà consulter cet article de blog sur le sujet : <http://andrew-stewart.ca/2012/10/31/vim-ctags> (en anglais).

Les plugins avancés

J'aurais pu faire un livre entier qui recense les plugins *Vim*, mais je pense que l'intérêt aurait été assez limité. Je ne vais donc pas vous décrire plus en détails d'autres plugins. En revanche je vous donne ci-dessous une liste de plugins qui pourraient vous intéresser. Cette liste est issue d'un sondage que j'avais effectué sur Twitter deman-

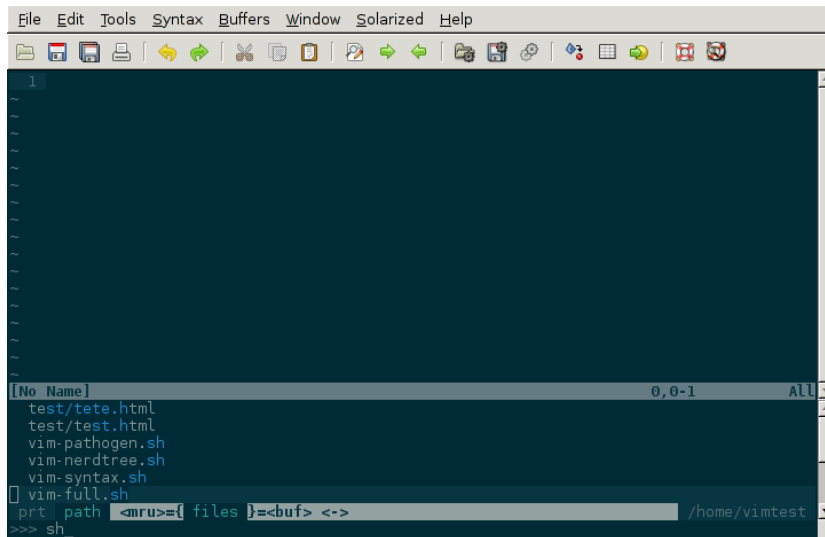


FIGURE 22: Vim avec CtrlP de lancé.

dant à mes followers quels étaient les plugins *Vim* indispensables selon eux. La voici :

- **neocomplcache**. C'est un plugin de complétion automatique. Il peut compléter les noms de fichiers, les attributs du langage que vous utilisez, les snippets et encore bien d'autres choses. Le repo Github : <https://github.com/Shougo/neocomplcache>.
- **surround**. Ce plugin permet de gérer (changer, ajouter, supprimer) tout ce qui « entoure » : les parenthèses, les crochets, les guillemets ... Par exemple vous pourrez en une combinaison de touches changer "Hello world !" en 'Hello world !' ou <q>Hello world!</q>. Le repo Github : <https://github.com/tpope/vim-surround>.
- **fugitive**. Si vous travaillez sur du code source vous utilisez forcément un gestionnaire de version de code source. Si ce n'est pas le cas vous pouvez aller vous cacher. Sinon si vous utilisez Git, Le plugin fugitive est pour vous. Il permet de gérer git directement dans *Vim*. Le repo Github : <https://github.com/tpope/vim-fugitive>
- **syntastic**. Syntastic vérifie pour vous la syntaxe de votre code source. Il va, comme peut le faire Eclipse par exemple, vous afficher vos erreurs de syntaxe directement dans *Vim*. Peut vous faire gagner en temps certain si vous éditez souvent du code. Le repo Github est par ici : <https://github.com/scrooloose/syntastic>
- **ctags + ctrlp**. Ctags est un petit programme externe qui va parcourir votre code source et qui va ensuite vous permettre de « suivre » vos fonctions dans votre code source. Très pratique

pour naviguer dans votre code source. Utilisé conjointement avec **ctrlp** décrit plus haut, il s'avère vite indispensable. Tout est expliqué ici : <http://andrew-stewart.ca/2012/10/31/vim-ctags>.

Pense-bête et exemples

Nous venons de faire un tour d’horizon de tout ce qui est nécessaire pour bien commencer dans la vie avec *Vim*. Tout cela devrait être suffisant pour pouvoir l’utiliser au quotidien. C’est le secret de la réussite avec *Vim* : réussir à l’encre dans nos habitudes journalières. Une fois que cela est fait, le reste devrait couler de source.

Cette dernière partie est là pour vous donner un endroit de référence où vous pourrez revenir comme bon vous semble lorsque vous serez un peu perdu sur comment faire telle ou telle chose avec *Vim*. Ce chapitre est composé de deux parties. La première est un ensemble de questions réponses qui couvre les principaux problèmes que les débutants rencontrent lorsqu’ils commencent. Le but est de répondre aux questions du type : « rha mais comment on fait ça, c’était pourtant si simple avec mon ancien éditeur ». La seconde partie est une liste (non exhaustive) des commandes *Vim* les plus utiles dont vous pourrez vous servir comme pense-bête. Allez hop, au boulot.

Questions / réponses

Comment quitter Vim ?

La première chose à faire est de se mettre en mode normal. Grosso modo, excitez-vous sur la touche **Esc** (**Échap**) ou la touche **;** en fonction de votre configuration et vous devriez vous retrouver en mode normal. Ensuite tapez **:q** pour quitter. Il y a de grandes chances que *Vim* ne vous laisse pas faire. Si vous avez des modifications non enregistrées par exemple, il ne voudra pas quitter. Vous pouvez annuler les modifications en le forçant à quitter grâce à l’utilisation de **!** comme ceci : **:q!**. Vous pouvez aussi enregistrer vos modifications puis quitter comme ceci : **:wq**.

Comment sauvegarder sous ?

En mode normal, si vous tapez `:w`, Vim par défaut sauvegarde vos modifications dans le fichier courant. Si vous souhaitez utiliser un autre nom de fichier pour « sauvegarder sous », vous avez juste à lui spécifier le nom du fichier après `w` comme ceci : `:w monfichier.txt`. Vim sauvegardera alors votre fichier sous le nom *monfichier.txt*. En revanche Vim n'ouvrira pas *monfichier.txt*, il restera sur votre précédent fichier.

Si vous souhaitez que Vim sauvegarde sous *monfichier.txt* et ouvre ensuite ce fichier dans le tampon courant, vous devrez utiliser `:sav monfichier.txt`.

Comment copier/couper coller ?

Celle là est facile, j'y ai déjà consacré un chapitre, cf. [Se déplacer par l'exemple : Essayer de copier / coller](#).

En résumé :

- Passez en mode visuel avec la touche **v**,
- Sélectionnez ce que vous voulez copier en vous déplaçant,
- Copiez avec la touche **y** ou couper avec la touche **x** ou la touche **d**,
- Collez après l'emplacement du curseur avec la touche **p** ou avant l'emplacement du curseur avec la touche **P**.

Comment créer un nouveau fichier ?

La façon traditionnelle de faire est de taper, en mode normal, `:e monfichier.txt` pour ouvrir un tampon (buffer) vide. Ensuite, sauvegardez votre tampon grâce à `:w`. Il sera sauvegardé sous le nom *monfichier.txt* dans le répertoire courant.

Vous pouvez aussi utiliser Lusty Explorer (cf. [Naviguer sur le disque et entre les fichiers : Lusty Explorer](#)) pour ce faire. Lancez le grâce à `,lr` ou `,lf`, tapez le nom du fichier que vous souhaitez créer puis appuyez sur la touche **Ctrl** (**Control**) puis en même temps la touche **e**. Vous pouvez ensuite le sauvegarder de la même manière que ci-dessus.

Annuler / Refaire

Pour annuler il suffit d'utiliser la touche **u** en mode normal. Pour annuler le annuler (donc refaire) maintenez la touche **Ctrl** (**Control**) appuyée puis la touche **r**.

*Pense-bête**Fichiers*

Résultat attendu	Action	Commentaire
Sauvegarder	<code>:w</code>	(w pour write)
Sauvegarder sous	<code>:w nomdefichier.txt</code>	Sauvegarde sous nomdefichier.txt mais n'ouvre pas nomdefichier.txt
Sauvegarder sous / ouvre	<code>:sav nomdefichier.txt</code>	Sauvegarde sous et ouvre nomdefichier.txt
Quitter sans sauvegarder (forcer à quitter)	<code>:q!</code>	
Sauvegarder et quitter	<code>:wq</code> (wq pour write and quit)	

Déplacements

Résultat attendu	Action
Se déplacer d'un caractère à gauche	<code>h</code>
Se déplacer d'un caractère en bas	<code>j</code>
Se déplacer d'un caractère en haut	<code>k</code>
Se déplacer d'un caractère à droite	<code>l</code>
Se déplacer à la fin d'un mot	<code>e</code>
Se déplacer au début d'un mot	<code>b</code>
Se déplacer au début du mot suivant	<code>w</code>
Se déplacer à la ligne 42	<code>:42</code>
Se déplacer au début du fichier	<code>gg</code> ou <code>:0</code>
Se déplacer à la fin du fichier	<code>GG</code> ou <code>:\$</code>
Se déplacer à la fin de la ligne	<code>\$</code>
Se déplacer au premier caractère non vide de la ligne	<code>^</code>
Se déplacer au début de la ligne	<code>0</code>
Descendre d'une page	<code>Ctrl+f</code>
Monter d'une page	<code>Ctrl+b</code>
Se déplacer à la première ligne de l'écran	<code>H</code>
Se déplacer au milieu de l'écran	<code>M</code>
Se déplacer à la dernière ligne de l'écran	<code>L</code>

Édition de texte

Résultat attendu	Action	Commentaire
Insérer avant le curseur	i	
Insérer avant le premier caractère non vide de la ligne	I	
Insérer après le curseur	a	
Insérer à la fin de la ligne	A	
Insérer une nouvelle ligne en dessous	o	
Insérer une nouvelle ligne au dessus	O	
Remplace le reste de la ligne	C	
Remplace un seul caractère (et reste en mode normal)	r	
Supprime le caractère après le curseur (comme la touche suppr.)	x	
Supprime le caractère avant le curseur (comme la touche backspace)	X	
Supprime la ligne courante	dd	
Copie la ligne courante	yy	
Colle après le curseur. Si c'est une ligne, colle la ligne en dessous.	p	
Colle avant le curseur. Si c'est une ligne, colle la ligne au dessus.	P	
Intervertit la case des caractères (majuscules / minuscules)	~	Marche en mode visuel
Déplace le texte vers la droite (indentation)	>	Marche en mode visuel
Déplace le texte vers la gauche	<	Marche en mode visuel
En mode visuel, supprime la sélection	d	Mode visuel
En mode visuel, remplace la sélection	c	Mode visuel
En mode visuel, copie la sélection	y	Mode visuel
Annuler (Undo)	u	
Refaire (Redo)	Ctrl+r	

Chercher et/ou remplacer

Résultat attendu	Action	Commentaire
Rechercher	<code>/toto</code>	Cherche la chaîne de caractères <i>toto</i> à partir de l'emplacement courant du curseur
Suivant	<code>n</code>	Affiche le prochain résultat de recherche
Précédent	<code>N</code>	Affiche le précédent résultat de recherche
Remplacer sur la ligne courante	<code>:s/toto/titi</code>	Remplace toto par titi sur la ligne courante (une fois)
Remplacer tout sur la ligne courante	<code>:s/toto/titi/g</code>	Remplace toto par titi sur la ligne courante (pour toutes les occurrences de toto)
Remplacer dans toutes les lignes	<code>:%s/toto/titi</code>	Remplace toto par titi sur toutes les lignes du fichier (une fois par ligne)
Remplacer tout dans toutes les lignes	<code>:%s/toto/titi/g</code>	Remplace toto par titi sur toutes les lignes du fichier (pour toutes les occurrences de toto par ligne)
Remplacer sur la ligne courante en ignorant la casse	<code>:s/toto/titi/i</code>	Remplace toto par titi sur la ligne courante (une fois)
Remplacer tout sur la ligne courante en ignorant la casse	<code>:s/toto/titi/gi</code>	Remplace toto par titi sur la ligne courante (pour toutes les occurrences de toto)