VINCENT JOUSSE

VIM - LE GUIDE POUR LES ÊTRES HUMAINS

VINCENT JOUSSE

PUBLIÉ PAR VINCENT JOUSSE

STYLE LaTeX http://tufte-latex.googlecode.com

Si vous n'avez pas payé cette copie, bah tant pis pour moi ;) Mais sachez que vous auriez dû !

Table des matières

Table des figures

Liste des tableaux

Merci à ma femme et mes enfants qui ont permis à ce livre de voir le jour.

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 actuellement[3]. 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 :

1. http://www.gnu.org/software/emacs/
2. http://www.vim.org/
3. À noter que *Vim* n'est arrivé qu'en 1991

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).

Enlarge 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 changerez de langage de programmation, vous n'aurez pas à changer d'éditeur. À noter que ce livre a bien sur 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 à ce manque en vous guidant tout au long de votre découverte de *Vim*[4].

Si vous aussi vous en avez marre d'attendre la release de TextMate 2 ? D'essayer le n-ième éditeur à la mode et de devoir tout réapprendre et ce jusqu'à la prochaine mode ? De devoir changer

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 ...

d'éditeur quand vous passez de votre Mac, à votre Linux, à votre serveur ? Vous aussi, rejoignez la communauté des gens heureux de leur éditeur de texte. Le changement, c'est maintenant.

Pour qui ?

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

Par exemple vous êtes :

Étudiant Si vous voulez faire bien sur un CV, c'est un must (en plus d'être un attrape geekette en puissance [5]). 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 LaTeX, vos présentations, 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 LaTeX).

5. À confirmer.

Prof 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 vous êtes pendable, 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 LaTeX, *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 votre outil**, comme un artiste, 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 donc Mac Os X aussi).
- 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*. 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 connaitre 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.

Le plus dur, c'est de commencer (et de continuer à commencer)

Alors, prêt pour l'aventure ? Prêt à sacrifier une heure pour débuter 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 !

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 "Mainstream" 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 look 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 Nerdtree. À savoir qu'à la fin de ce livre, vous n'aurez plus besoin de la souris (et donc des menus et autres boutons).

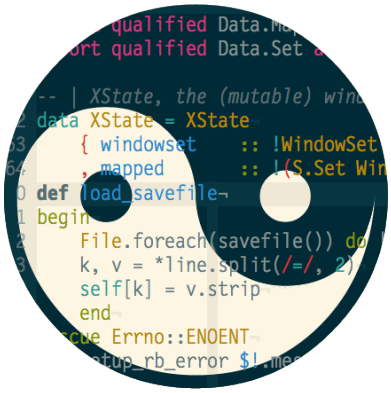

Figure 1: Le thème *Solarized* en sombre et en clair. `http://ethanschoonover.com/solarized`

Ce chapitre est indispensable si vous n'avez que peu d'expérience

(voir 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 sur 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 MacVim [6] sans aucune hésitation. Si vous utilisez GNU/Linux ou tout autre système "Unix" vous devriez surement 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 sure le site officiel de *Vim* [7], 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.

6. MacVim : `http://code.google.com/p/macvim/`

7. Page de téléchargement officielle de *Vim* : `http://www.vim.org/download.php`

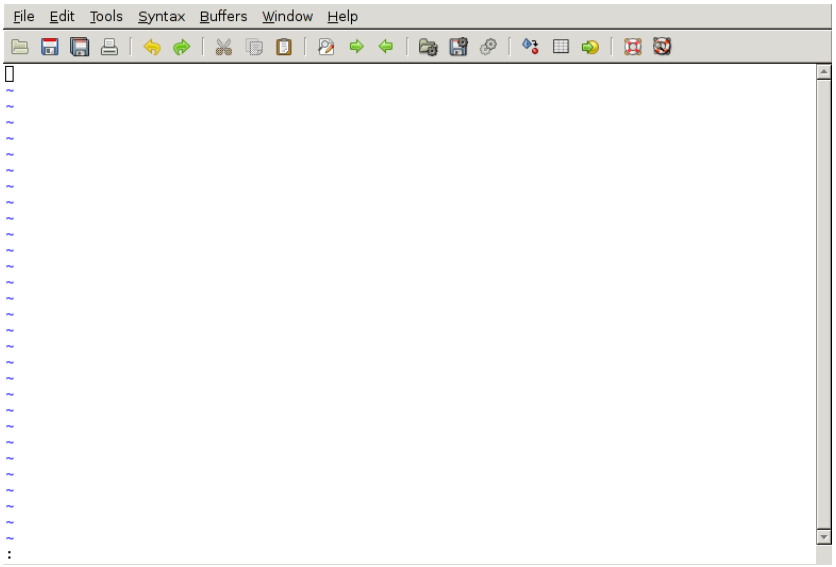

FIGURE 2: Nouveau fichier vide.

Commençons par entrer un commentaire dans d'entête 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 [8].

```
" 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.

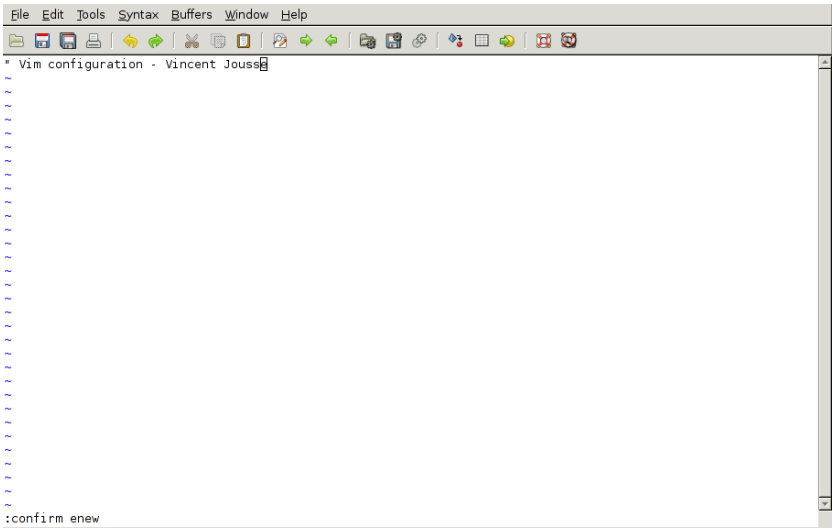

FIGURE 3: Mon premier commentaire.

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.

8. 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*

Les modes : d'où Vim *tire sa puissance*

Je pense que nous serons tous à peu prêt 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 sur, 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 et les troubles musculosquelettiques en général [9]. D'après *Wikipedia*, c'est le type de maladie professionnelle la plus courante à l'heure actuelle [10].

Vous oubliez aussi le mouvement de votre main droite pour vous déplacer vers les touches directionnelles afin de bouger votre curseur. 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 de claviers des marques sur les touches F et J, c'est pour vous donner un repère tactile d'où doivent se trouver vos indexes dans la position de repos.

9. 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.
10. https://fr.wikipedia.org/wiki/Troubles_musculosquelettiques

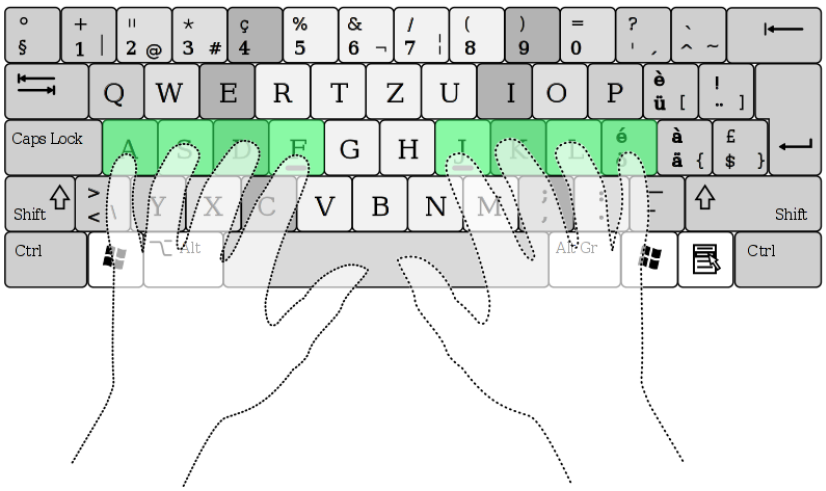

FIGURE 4: Position de repos, clavier QWERTY. *Illustration par Cy21 - CC-BY-SA-3.0 (www.creativecommons.org/licenses/by-sa/3.0) ou GFDL (www.gnu.org/copyleft/fdl.html), via Wikimedia Commons http://commons.wikimedia.org/wiki/File:Typing-home-keys-hand-position.svg*

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 [11] (copier/coller, macros, ...), tantôt à sélectionner [12] et tantôt à insérer du texte [13]. Tout cela bien sur en évitant l'utilisation de com-

11. C'est le mode *normal*
12. C'est le mode *visuel*
13. C'est le mode *insertion*

binaisons 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 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'autre moyens de faire. Par exemple pour rentrer en mode insertion tout en créant une nouvelle ligne en dessous de la ligne courante (peut importe où se trouve votre curseur sur la ligne), on utilisera la touche o en mode *normal*.

SI VOUS VOULEZ POUSSER LA DÉMARCHE jusqu'au bout, vous pouvez aussi vous procurer un clavier orthogonal *TypeMatrix* [14]. C'est ce que j'utilise personnellement, et mes doigts m'en remercient tous les jours.

14. 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* [15] 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.

Listing 2: Une configuration par défaut sensée.

15. 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 jute "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`

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.

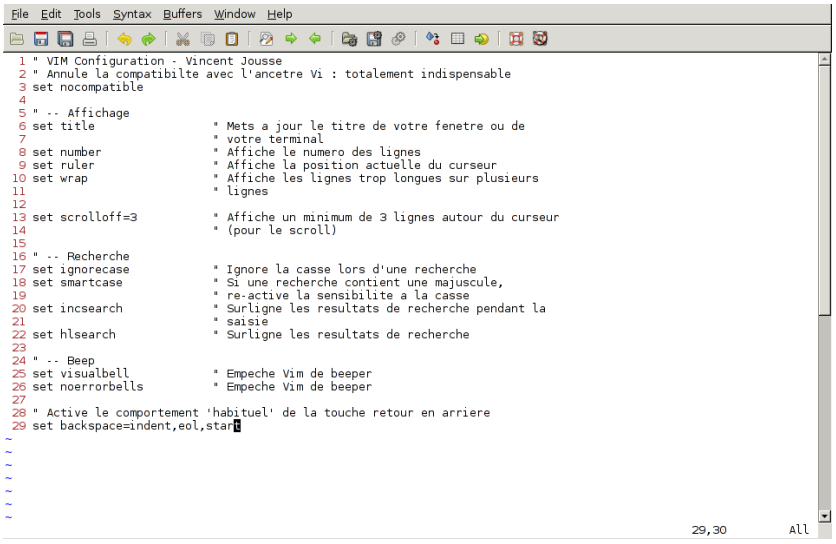

Figure 5: *Vim* après votre première configuration.

Bon c'est bien beau tout ça mais ça manque un peu de couleurs. 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 là fin de votre fichier de configuration *.vimrc*.

```
" Active la coloration syntaxique
syntax enable
" Active les comportements specifiques 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 6 [16]. C'est une bonne première étape, passons maintenant à l'utilisation d'un thème.

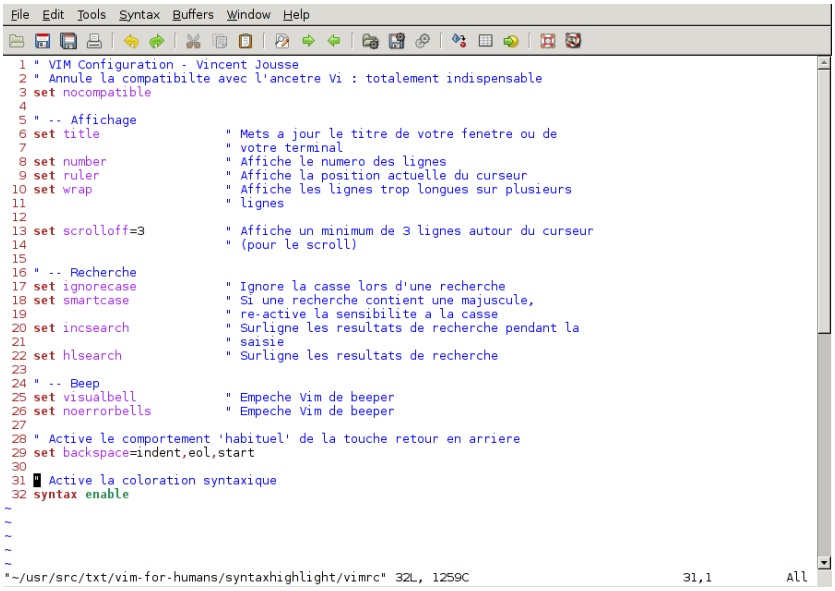

FIGURE 6: Coloration syntaxique par défaut.

16. Pour l'instant, le plus facile pour que les modifications apportées à votre *.vimrc* soient prisent 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` étant un raccourci pour `:source`.

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` [17] au même endroit que votre *.vimrc* [18]. Dans ce répertoire `.vim`, créez un sous-répertoire nommé `colors`. Téléchargez

17. 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
18. Dans votre répertoire utilisateur donc.

ensuite le fichier du thème Solarized `https://raw.github.com/altercation/vim-colors-solarized/master/colors/solarized.vim` [19](c'est le même fichier pour les deux versions du thème) et copiez le dans le répertoire `vim/colors/` fraichement créé. 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` pour la valeur `background`.

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

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

Listing 4: Activation de la coloration syntaxique.

Les images 7 et 8 vous donnent un aperçu des deux variantes (ma préférence allant à la variante sombre soit dit en passant).

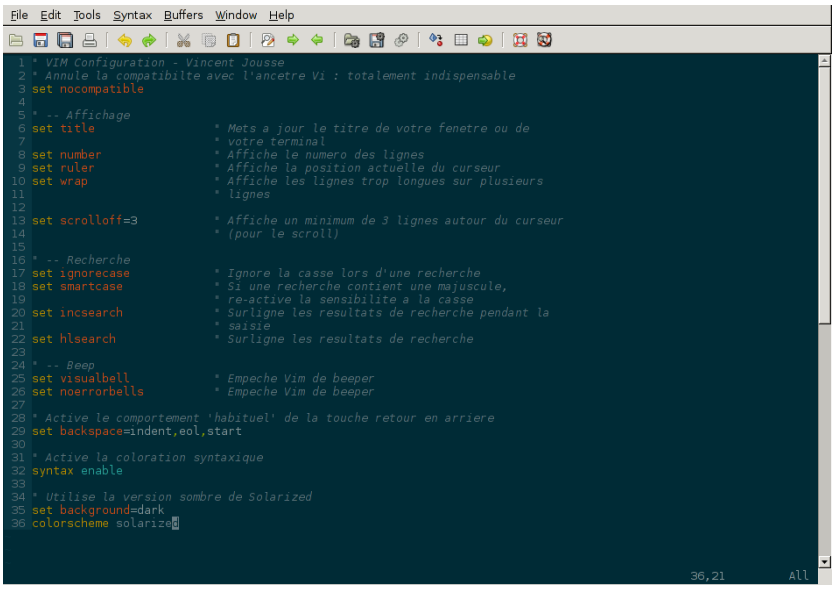

FIGURE 7: Le thème Solarized sombre.

UN BONUS (si vous n'utilisez pas *Vim* directement dans votre terminal) serait de choisir une police de caractère qui vous convient un peu mieux, c'est bien sur facultatif mais bon, je présume que certains d'entre vous sont des esthètes aguerris.

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

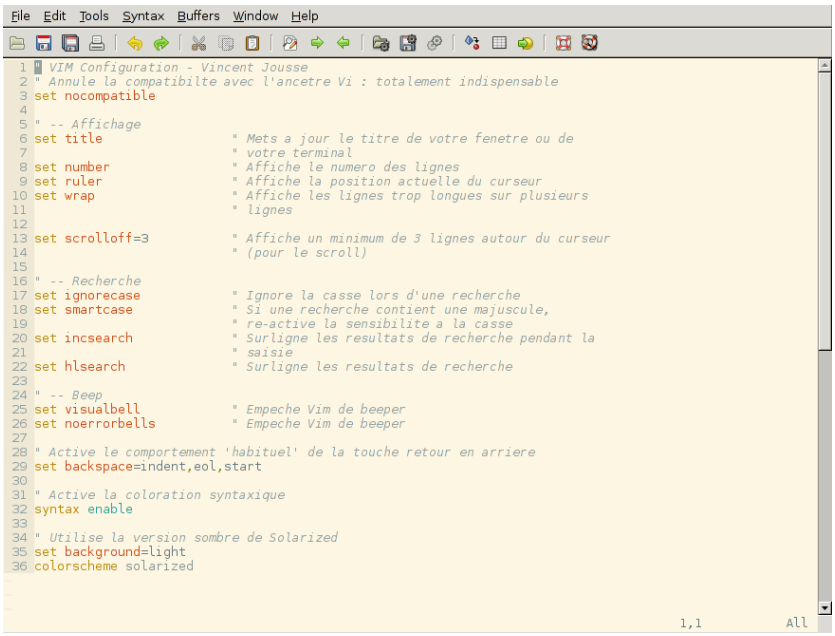

FIGURE 8: Le thème solarized clair.

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

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

Vous pouvez bien sur 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 sur changer la taille de la police si vous le souhaitez. Pour avoir la liste des polices disponibles tapez on 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 plugin pour éviter que ça devienne trop le bazaar dans vos plugins, puis installer ensuite le plugin qu'il nous faut pour explorer un répertoire de fichiers.

Gestionnaire de plugins : Pathogen

Pathogen [20] est typique du plugin que vous découvrez après avoir commencé à configurer votre *Vim* et qui génère ce type de réaction : "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 plus part 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 la que pathogen arrive à la rescousse, il va vous permettre d'installer chaque plugin dans un sous répertoire rien que pour lui. La figure 9 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 [21].

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.github.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 [22].

20. `https://github.com/tpope/vim-pathogen/`

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

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

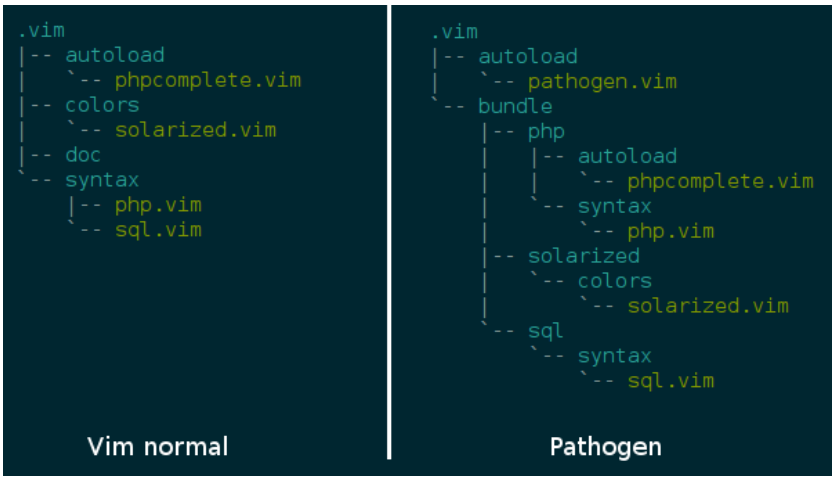

FIGURE 9: *.vim* avant et après Pathogen.

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

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

Listing 7: Installation de pathogen.

Nous installerons ensuite nos plugins 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 plugins.

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 soit même, nous allons ranger notre petit plugin solarized en utilisant pathogen. Il nous suffit de créer un répertoire `solarized` dans notre répertoire `bundle` fraichement créé [23]. Nous déplaçons ensuite le répertoire `colors` dans le répertoire `solarized` cf. le listing 10.

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

```
# 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 près 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 lisse.

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 (par défaut) dans 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 le plugin *Ctrl-p* 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 adresse [24] : 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.

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