

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 totalement **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, Sublime Text, Notepad++ ou NetBeans proposent, c'est à dire un environnement à minima utilisable tel quel, même si l'on n'en exploite pas la totalité.

Voici donc ce qui manque à un *Vim* nu (et ce qui est, de mon point de vue, 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'écrire ou de se procurer 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

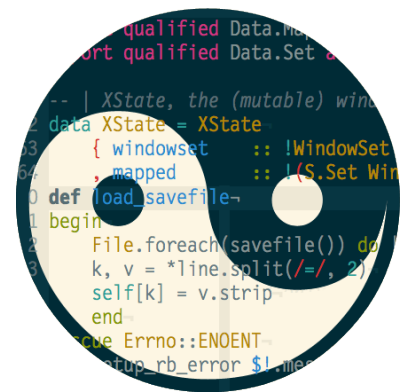


FIGURE 1: Le thème *Solarized* en sombre et en clair. <http://ethanschoonover.com/solarized>

la souris (et donc des menus et autres boutons).

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*. 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 devriez certainement commencer par installer une version de *Vim*. Si vous utilisez un Mac, essayez MacVim⁵ 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*⁶, mais je ne l'ai pas testée.

Au lancement de *Vim*, vous devriez avoir un texte d'accueil vous encourageant à aider les pauvres enfants en Ouganda. Ce texte disparaîtra dès que nous allons saisir des caractères dans *Vim*. Nous allons commencer 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). Le texte d'accueil par défaut de *Vim* devrait avoir disparu et vous devriez avoir une page blanche qui ressemble plus ou moins à la figure 2.

Vous devriez maintenant pouvoir entrer le commentaire ci-dessous⁷.

```
" 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.

Tout ça pour ça me direz-vous, et vous avez bien raison. Et encore, on n'a même pas encore vu comment le sauvegarder. Mais tout cela a une logique que je vais vous expliquer. L'avantage de *Vim* est qu'il

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

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

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

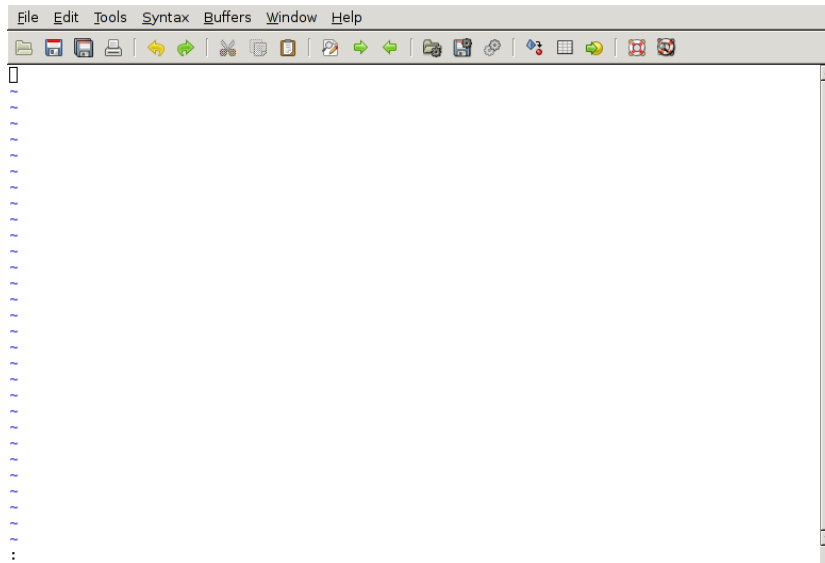


FIGURE 2: Nouveau fichier vide.

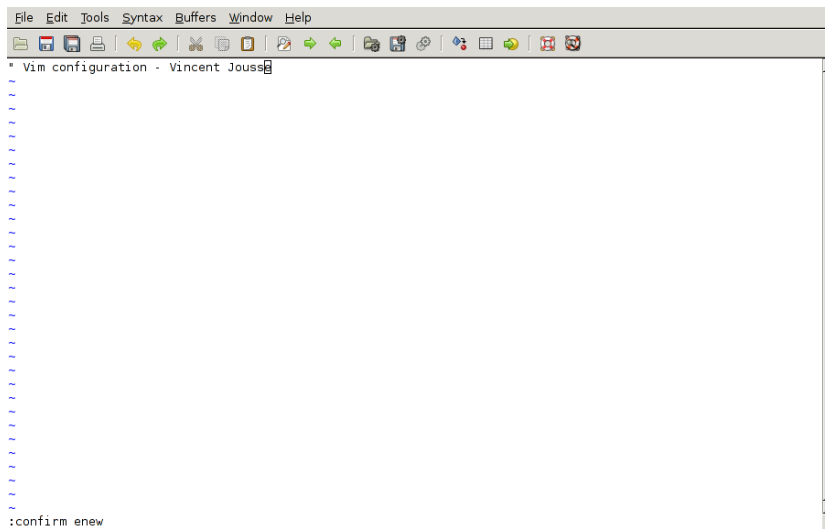


FIGURE 3: Mon premier commentaire.

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 *Insertion*) mais juste pour se déplacer et manipuler du texte. C'est la présence de ces deux 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 musculosquelettiques⁸. D'après *Wikipedia*, c'est le type de maladie professionnelle la plus courante à l'heure actuelle⁹.

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 tout ça ? 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 plupart 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.

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 texte¹⁰ (copier/coller, macros, ...), tantôt à sélectionner¹¹ et tantôt à insérer du texte¹². Tout cela bien sûr en évitant l'utilisation de com-

8. 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.

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

10. C'est le mode *normal*

11. C'est le mode *visuel*

12. C'est le mode *insertion*

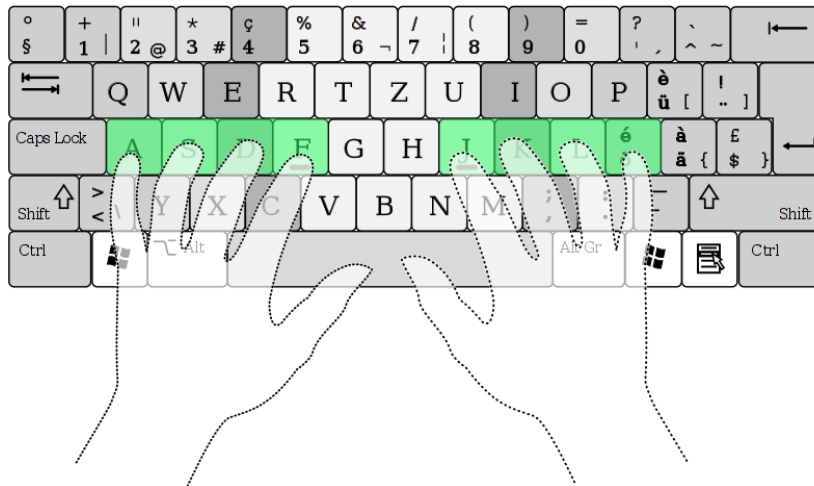


FIGURE 4: Position de repos, clavier QWERTY. Illustration par Cy21 - CC-BY-SA-3.0 (www.creativecommons.org/licenses/by-sa/3.0) ou GFDL (www.gnu.org/copyleft/fdl.html), via Wikimedia Commons <http://commons.wikimedia.org/wiki/File:Typing-home-keys-hand-position.svg>

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'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 recommanderais 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*¹³. 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/>

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

`comment-doubler-sa-vitesse-de-frappe-au-clavier/.`

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`¹⁴ 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 compatibilite avec l'ancetre Vi : totalement indispensable
set nocompatible

" -- Affichage
set title           " Met a jour le titre de votre fenetre ou de
                    " votre terminal
set number          " Affiche le numero 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 sensibilite a la casse
set incsearch       " Surligne les resultats de recherche pendant la
                    " saisie
set hlsearch        " Surligne les resultats de recherche

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

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

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

Listing 2: Une configuration par défaut sensée.

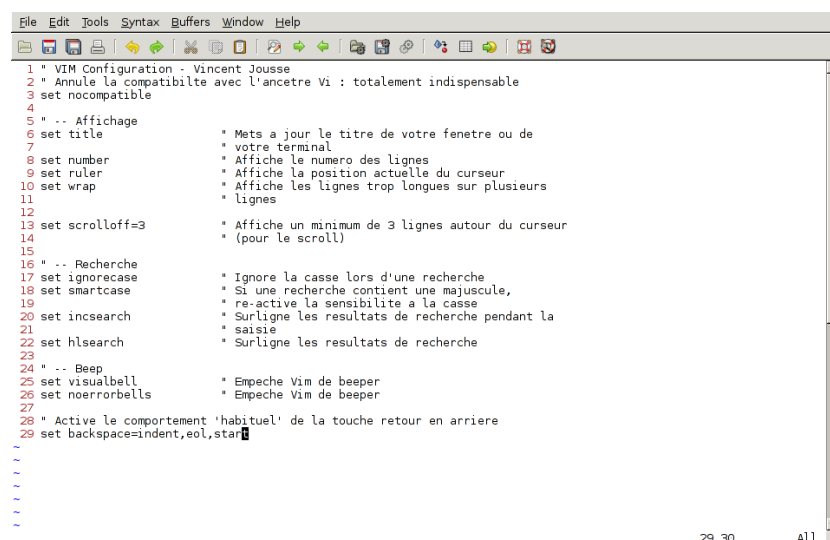
Pour ceux qui ont fait un copier/coller, il ne vous reste plus qu'à sauvegarder votre fichier nouvellement créé. Nous voulons le placer à la racine de votre compte utilisateur, c'est à dire l'enregistrer sous `~/.vimrc`. Sous Mac OS X et Linux, `~` désigne le répertoire d'accueil de l'utilisateur courant. Attention, les fichiers commençant par un `.` sont des fichiers cachés sous Linux et Mac OS X, ne vous étonnez donc pas de ne pas le voir par défaut dans votre navigateur de fichiers.

14. 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

Pour les utilisateurs de la souris, il suffira de se servir du menu Fichier (ou File) puis Sauvegarder sous (ou Save as) et de l'enregistrer dans le répertoire d'accueil de l'utilisateur courant sous le nom de `.vimrc`. Pour ceux qui veulent déjà utiliser le clavier, il vous suffira, après avoir appuyé sur la touche `Esc` (Échap) pour repasser en mode *Normal*, de taper `:sav /.vimrc`. Pour sauvegarder vos prochaines modifications, utilisez le menu avec la souris ou tapez en mode *Normal* `:w`.

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://vimebook.com/link/fr/firstconfig>.

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.



```

1  " VIM Configuration - Vincent Jousse
2  " Annule la compatibilité avec l'ancêtre Vi : totalement indispensable
3  set nocompatible
4
5  " -- Affichage
6  set title           " Mets à 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 sensibilité à 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

```

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 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 6¹⁵. C'est une bonne première étape, passons maintenant à l'utilisation d'un thème.

```
File Edit Tools Syntax Buffers Window Help
1 " VIM Configuration - Vincent Jousse
2 " Annule la compatibilité avec l'ancêtre Vi : totalement indispensable
3 set nocompatible
4
5 " -- Affichage
6 set title           " Mets à jour le titre de votre fenêtre ou de
7                     " votre terminal
8 set number         " Affiche le numéro 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 sensibilité à la casse
20 set incsearch      " Surligne les résultats de recherche pendant la
21                   " saisie
22 set hlsearch       " Surligne les résultats de recherche
23
24 " -- Beep
25 set visualbell     " Empêche Vim de beeper
26 set noerrorbells  " Empêche Vim de beeper
27
28 " Active le comportement 'habituel' de la touche retour en arrière
29 set backspace=indent,eol,start
30
31 " Active la coloration syntaxique
32 syntax enable
~
~
~
~/usr/src/txt/vim-for-humans/syntaxhighlight/vimrc" 32L, 1259C 31,1 All
```

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`¹⁶ au même endroit que votre `.vimrc`¹⁷. Dans ce répertoire `.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> (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.

Activez ensuite le thème *Solarized* dans votre `.vimrc` comme le montre le code dans le listing 4. Pour tester le thème clair, rem-

15. 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` ou `:so $MYVIMRC`. `:so` étant un raccourci pour `:source`.

FIGURE 6: Coloration syntaxique par défaut.

16. 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

17. Dans votre répertoire utilisateur donc.

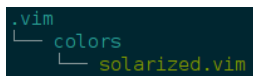


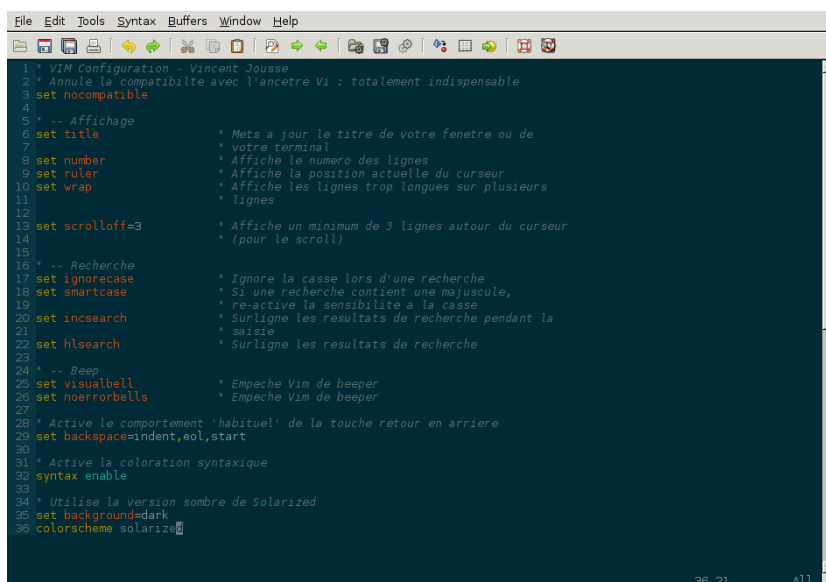
FIGURE 7: Le contenu du répertoire .vim avec Solarized.

placez `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).

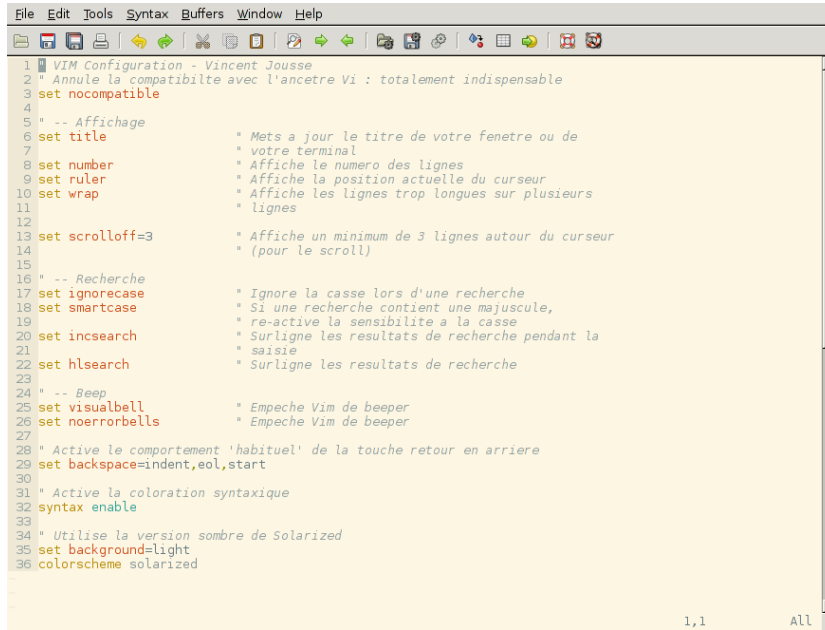
FIGURE 8: Le thème *Solarized* sombre.

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 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 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.



```

1 | VIM Configuration - Vincent Jousse
2 | Annule la compatibilité avec l'ancêtre Vi : totalement indispensable
3 | set nocompatible
4 |
5 | -- Affichage
6 | set title           " Mets à jour le titre de votre fenêtre ou de
7 |                   " votre terminal
8 | set number         " Affiche le numéro 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 sensibilité à la casse
20 | set incsearch      " Surigne les résultats de recherche pendant la
21 |                  " saisie
22 | set hlsearch       " Surigne les résultats de recherche
23 |
24 | -- Beep
25 | set visualbell     " Empêche Vim de beeper
26 | set noerrorbells  " Empêche Vim de beeper
27 |
28 | Active le comportement 'habituel' de la touche retour en arrière
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 solarized

```

FIGURE 9: Le thème Solarized clair.

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 <http://vimebook.com/link/fr/syntaxhlconfig>. Je ne m'attarderai pas plus sur les polices, c'est assez dépendant de votre système d'exploitation, et un peu moins de Vim.

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 le plugin adéquat pour explorer un répertoire de fichiers.

Gestionnaire de plugins : Pathogen

*Pathogen*¹⁸ est le genre de plugin typique 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 sur la manière d'installer et de configurer 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 suite¹⁹.

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'installer²⁰.

```
# 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
```

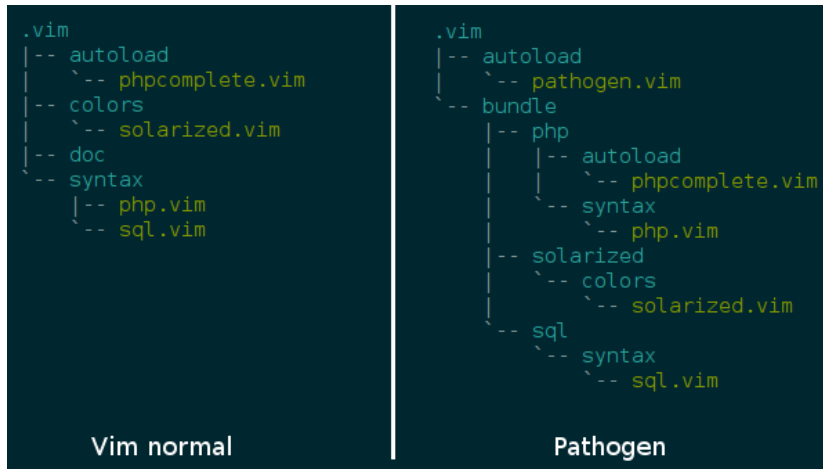
Listing 7: Installation de pathogen.

Nous installerons ensuite nos plugins directement dans le ré-

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

19. 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.

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

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

pertoire `.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`. Il est impératif de placer le code **au début** de votre fichier `.vimrc`, sinon ça ne marchera pas.

```
" 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 créé²¹. 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.

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

Actuellement, Pathogen reste encore le gestionnaire de plugins Vim le plus utilisé. Mais depuis peu, un challenger est arrivé, il s'appelle Vundle <https://github.com/gmarik/vundle>. J'ai choisi de vous présenter Pathogen car c'est de lui que vous entendrez parler le plus, mais sachez que Vundle est aussi une alternative intéressante : il est compatible avec Pathogen et il gère les versions et les mises à jours de vos plugins directement depuis internet. Pour ceux qui connaissent Ruby, c'est le Bundler²² pour Vim.

22. <http://gembundler.com/>

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 (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 [Les plugins indispensables](#) 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 que vous pouvez télécharger à cette adresse : 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` :

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

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

C'est, j'en conviens, une commande un peu barbare qui pourrait se traduire en bon vieux français par : à chaque ouverture de vim (`vimenter`), peu importe le type de fichier (*), lancer *The NERD Tree* (`NERDTree`).

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` puis `w`²³. Ce raccourci clavier sera d'ailleurs toujours valable pour naviguer entre vos différentes fenêtres *Vim* (il n'est pas spécifique à *The NERD Tree*).

23. La touche *Control* (Ctrl) et tout en la laissant appuyée la touche `w`. Vous pouvez ensuite tout lâcher pour appuyer une nouvelle fois sur `w`.

Nous voilà fin prêts

Voilà, vous avez fait le plus dur. Enfin presque. Nous venons de couvrir ce qui 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

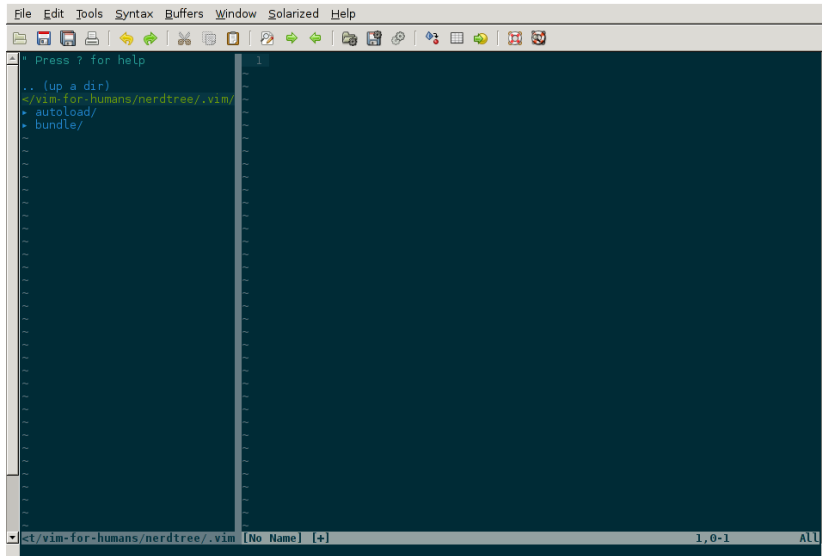


FIGURE 11: `.vim` avec *The NERD Tree* d'activé.

encore toutes les possibilités. Je vous recommande à ce stade de commencer à l'utiliser dans votre vie quotidienne. N'hésitez pas à user et à abuser de la souris et des différents menus qui sont à votre disposition. Le but ici étant de réduire l'impact de l'utilisation de *Vim* sur votre travail quotidien. Ce n'est pas encore le temps de briller en société. Vous apprendrez les raccourcis clavier au fur et à mesure, et ça ne fait pas de vous un utilisateur de *Vim* de seconde zone. Il faut bien commencer un jour.

Nous allons maintenant aborder ce qui fait l'unicité de *Vim* : 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 convient²⁴. C'est vous qui voyez !

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