Chapitre 2

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* (http://www.gnu.org/software/emacs/), de *Vi* et de son « successeur amélioré » *Vim* (http://www.vim.org/). Ils ont été créés dans les années 70 et sont toujours très utilisés actuellement. 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 (même si je dois avouer que le site d'*Emacs* a fait des efforts depuis la première version de ce livre). Voici quelques **raisons de leur succès** :

Ils s'apprennent 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.

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

Ils augmentent votre vitesse d'édition de texte
: Grâce à leurs particularités (notamment l'utilisation du clavier), ils permettent d'éditer du texte très rapidement.

Ce sont de vrais 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 sûr été écrit avec *Vim*.

Et pourtant, ces éditeurs de texte 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*. Je laisse *Emacs* à ceux qui savent (pour un bref comparatif c'est ici : https://fr.wikipedia.org/wiki/-Guerre_d"%C3%A9diteurs. Les goûts et les couleurs …). Il ne prétend pas être un guide exhaustif, vous pouvez essayer *A Byte of vim* (en anglais) pour celà : https://vim.swaroopch.com/. En revanche, il prétend diminuer la marche à franchir pour s'habituer à *Vim*. À mon sens, le plus compliqué avec *Vim*, c'est de ne pas se décourager de suite et de trouver une méthode qui vous permette de l'apprendre au fur et à mesure. Nous avons tous un travail à effectuer au quotidien, et perdre toute sa productivité à cause d'un changement d'éditeur de texte n'est pas envisageable.

Vous trouverez beaucoup de personnes pour vous dire « mais tu n'as qu'à t'y mettre pour de bon », « tu verras après ça va aller mieux », certes, mais vous devez toujours être productif au jour le jour, ça ne change rien au problème. L'approche de ce livre est la suivante :

— Disposer d'un *Vim* un temps soit peu moderne : coloration syntaxique et jolies couleurs.
— Pouvoir utiliser *Vim* comme n'importe quel autre éditeur de texte : éditer facilement du code et naviguer entre les fichiers en utilisant la souris.
— Apprendre des raccourcis clavier et se passer de la souris au fur et à mesure.
— Installer un *best-of* des plugins pour commencer à tirer partie de la puissance de *Vim*.

À partir du point numéro 2, vous pourrez déjà utiliser *Vim* au quotidien sans perdre énormément de productivité. Et c'est là que la différence se fait : si vous pouvez intégrer *Vim* dans votre quotidien c'est gagné. Vous aurez ensuite toute votre vie pour connaître tous les raccourcis clavier.

Vous aussi vous en avez marre d'attendre la release de TextMate 2 (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) ? 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 ? De ne pas pouvoir utiliser VSCode sur votre serveur ou de le trouver trop lourd ? Alors vous aussi, rejoignez la communauté des gens heureux de leur éditeur de texte. **Le changement, c'est maintenant** !

2.1 Et Neovim ?

Petit aparté au sujet de *Neovim* https://neovim.io/ (si vous ne savez pas ce que c'est, vous pouvez sauter cette partie). J'ai pris le parti de me concentrer uniquement sur *Vim* dans ce livre afin de ne pas frustrer les personnes ne pouvant utiliser que *Vim*.

Si vous êtes un utilisateur de *Neovim*, tout ce qui est dans ce livre reste valable, *Neovim* étant compatible avec *Vim*. Qui plus est, les modes, les manipulations de texte, et tout ce qui ne concerne pas les plugins est commun à *Vim* et à *Neovim* : les apprendre pour *Vim* ou pour *Neovim* ne fait aucune différence. Vous pouvez d'ailleurs suivre ce livre en utilisant *Neovim* à la place de *Vim* sans aucun souci.

La particularité de *Neovim* réside dans le fait qu'il est plus activement maintenu que *Vim* et entre autres qu'il utilise le langage de programmation **Lua** pour la gestion des plugins. Cela a pour conséquence que les plugins écrits pour *Neovim* ne sont pas compatibles avec *Vim* (l'inverse n'étant pas vrai, les plugins écrits pour *Vim* sont compatibles avec *Neovim*). Une partie de la communauté de *Vim* est passée à *Neovim* et les plugins pour *Neovim* fleurissent bien plus vite que ceux pour *Vim*. Pour le but de ce livre, cela ne fera pas de différence et vous pouvez le suivre sans aucun problème avec *Vim* ou *Neovim*. Je ferai sûrement d'autres contenus spécifiques à *Neovim* dans le futur.

2.2 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 sur un CV, c'est un must. C'est en effet un gage de sérieux de voir qu'un étudiant a pris sur son temps personnel pour apprendre *Vim*. De plus, 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, Markdown ou reStructuredText).

  Petit conseil d'ami, pour vos présentations, n'hésitez pas à utiliser un outil comme *impress.js* : https://github.com/impress/impress.js. Basé sur du HTML/JS/CSS, je vous le recommande grandement pour des présentations originales et basées sur des technologies non propriétaires. Il existe aussi *reveal.js* (https://lab.hakim.se/reveal-js/) et son éditeur en ligne *slid.es* (https://slid.es/).

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 ce que l'on pourrait appeler 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. Administrer un système Unix à distance est un des cas d'utilisation parfait pour *Vim* (un éditeur de texte surpuissant ne nécessitant pas d'interface graphique).

Écrivain
: Si vous écrivez en markdown/reStructuredText/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.

2.3 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.

2.4 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 de 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* (*Vim* est suffisant pour cela de toute façon). 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 parfait pour travailler.

2.5 Le plus dur, c'est de 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 ! 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 :

— Essayez de faire entrer *Vim* dans vos habitudes. Suivez le premier chapitre de ce guide jusqu'à la partie concernant l'explorateur de fichiers utilisable à la souris *vim-fern*. Ensuite, vous pourrez utiliser *Vim* comme un Notepad++ ou un TextMate ou un Sublime Text. Vous n'utiliserez que 1% des capacités de *Vim* mais peu importe. Ce qui est important, c'est de le faire entrer dans votre quotidien.

— Gardez une feuille avec les principaux raccourcis imprimée à côté de vous. Le but ce n'est pas de les apprendre par cœur, mais c'est de les avoir à portée de main quand vous vous demanderez « mais il y a certainement une façon plus efficace de faire cela ».
— Gardez la foi. Au début vous resterez un sceptique quant à l'utilité de tout réapprendre avec *Vim*. 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 complètement rentable de suite.

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

Chapitre 3

Rendre *Vim* utilisable

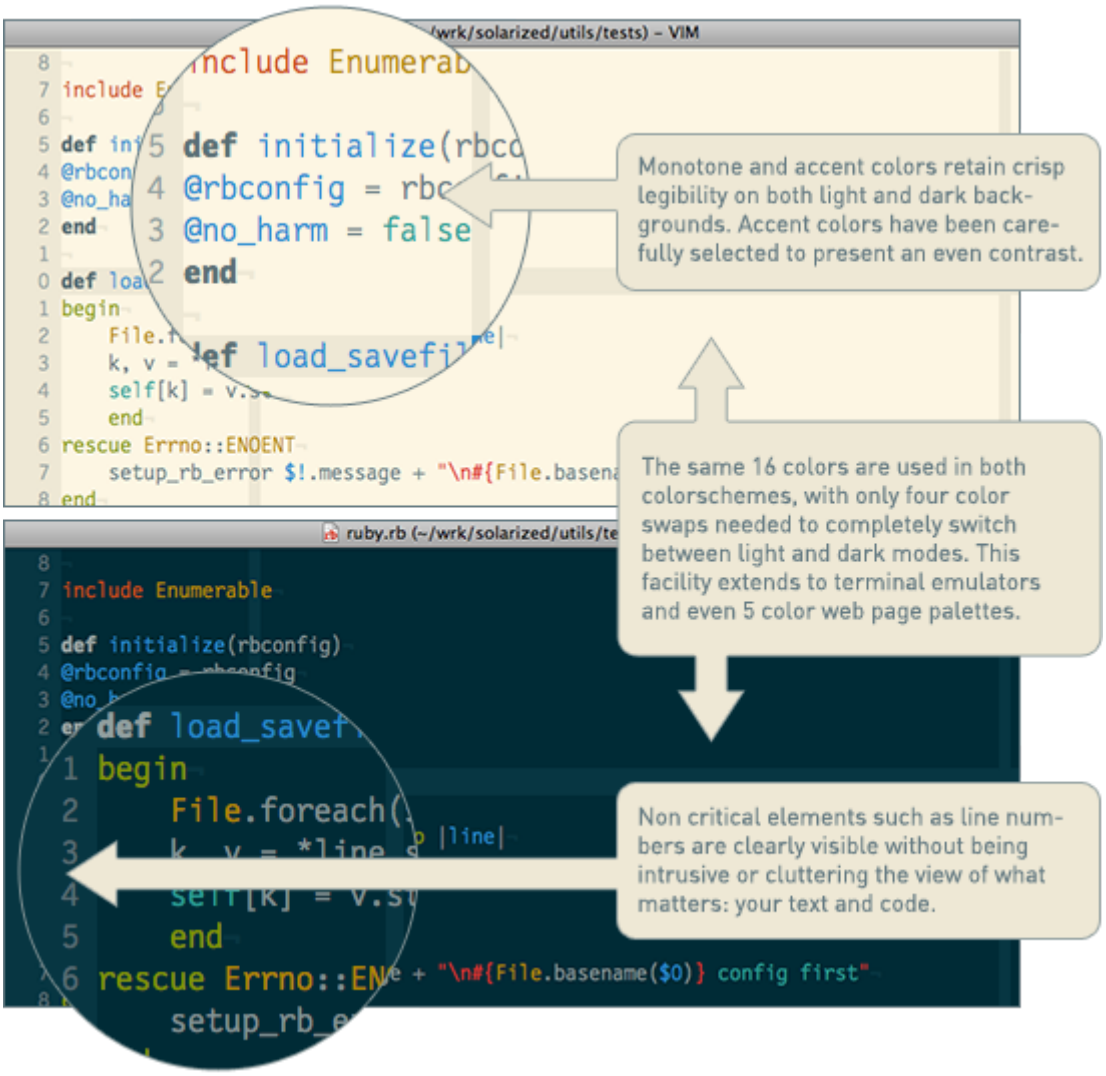
Ça peut paraître étonnant comme approche, mais c'est pour moi la première chose à faire : rendre *Vim* utilisable par un humain lambda. Si tout le monde semble s'accorder sur le fait que *Vim* est un **éditeur très puissant**, tout le monde pourra aussi s'accorder sur le fait que de base, il est totalement **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 VSCode, 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 belle image qui suit 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 *vim-fern* (https://github.com/lambdalisue/vim-fern). À savoir qu'à la fin de ce guide, vous n'aurez plus besoin de 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*).

3.1 Préambule indispensable : le mode insertion

Prenons le pari de créer le fichier `~/.vimrc` avec *Vim* lui-même. Comme je vous le disais, lus tôt vous commencerez, mieux ce sera. Vous devrez certainement commencer par installer une version de *Vim*. Si vous utilisez un Mac, essayez MacVim (https://macvim.org/) sans aucune hésitation. Si vous utilisez GNU/Linux ou tout autre système « Unix » vous devriez sûrement avoir la commande *vim* à disposition dans votre terminal. Vous pouvez aussi utiliser l'interface graphique *gVim* qui devrait être facilement installable grâce à votre gestionnaire de logiciels, pour Ubuntu le paquet correspondant est *vim-gnome*. Faites attention à bien installer la version **complète** de vim, notamment avec le support de Ruby et de Lua dont nous aurons besoin par la suite. Pour Ubuntu, le paquet sembler s'appeller *vim*. Pour Mac OS X, la question ne se pose pas, MacVim inclut déjà tout ce qu'il faut. Pour Windows, il semblerait y avoir une version disponible sur le site officiel de *Vim* (http://www.vim.org/download.php), mais je ne l'ai pas testée.

Pour ma part, j'utilise *vim* directement en ligne de commande, sous Archlinux, dans un terminal kitty avec la police Nerd Fonts FiraCode Nerd Font. C'est avec cette configuration que les capture d'écran de ce livre sont réalisées.

Au lancement de *Vim*, vous devriez avoir un texte d'accueil vous encourageant à aider les pauvres enfants en Ouganda. Ce texte disparaitra 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`. Vous devriez avoir *une page* qui ressemble plus ou moins à la figure ci-dessous, notez bien le `--INSERT--` en bas à gauche qui nous indique que nous sommes en mode insertion (le mode où nous pouvons saisir du texte). Pour information, le thème de mon terminal est un thème sombre, il est donc possible que pour l'instant, les couleurs de votre *Vim* soient différentes.

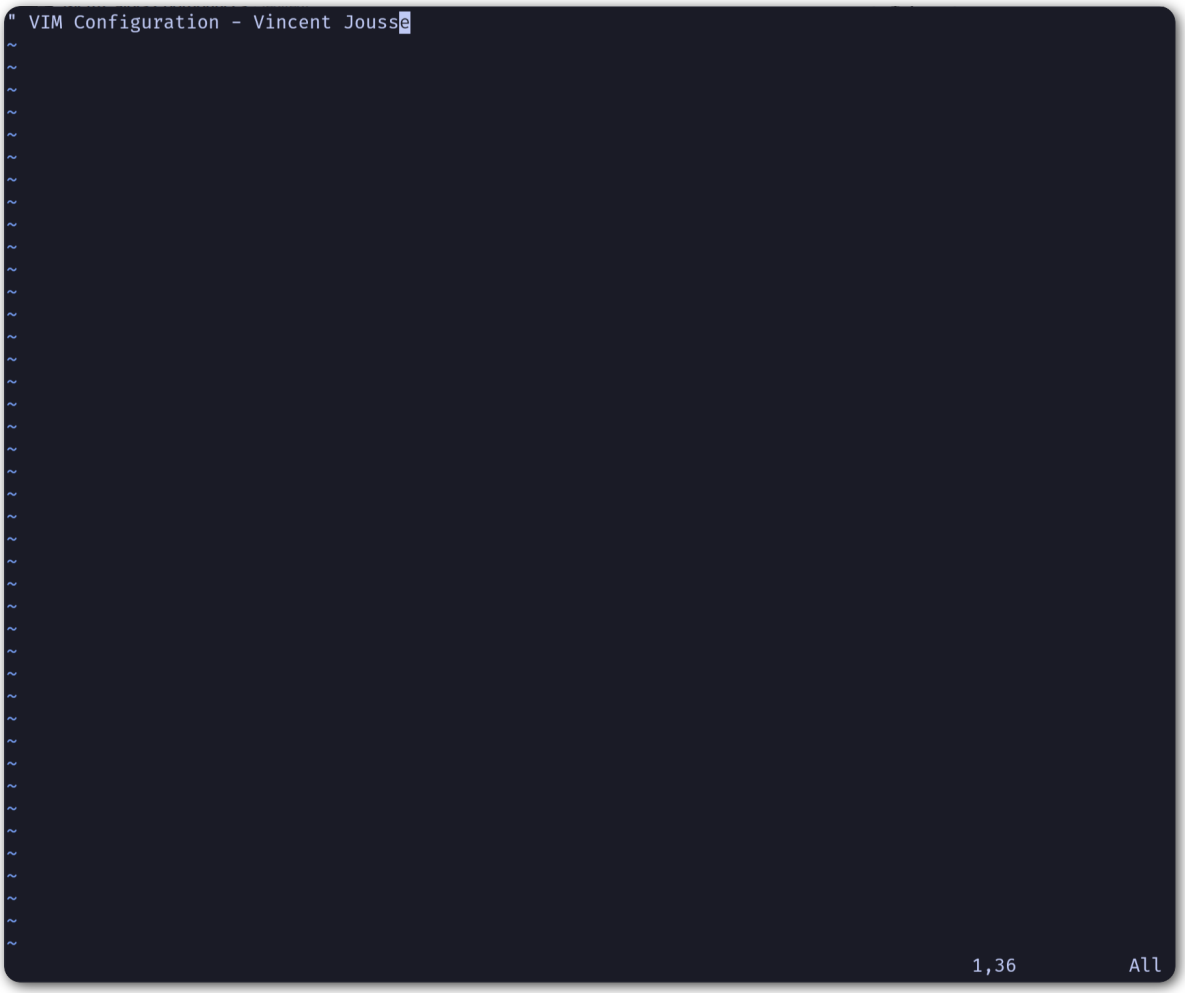
À l'écriture de ces lignes, la version de *Vim* que j'utilise est la `9.1.380`.

À noter : 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*.

Vous devriez maintenant pouvoir entrer *le commentaire ci-dessous* (je vous laisse évidemment changer mon nom par le votre 🤓).

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

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, comme vous pouvez le voir sur *la copie d'écran de Vim avec votre joli commentaire*.



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 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.

3.2 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 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 musculosquelettiques. 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. D'après *Wikipedia*, c'est le type de maladie professionnelle la plus courante à l'heure actuelle (https://fr.wikipedia.org/wiki/Troubles_musculosquelettiques).

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 *l'image ci-dessous avec la position idéale des mains*.

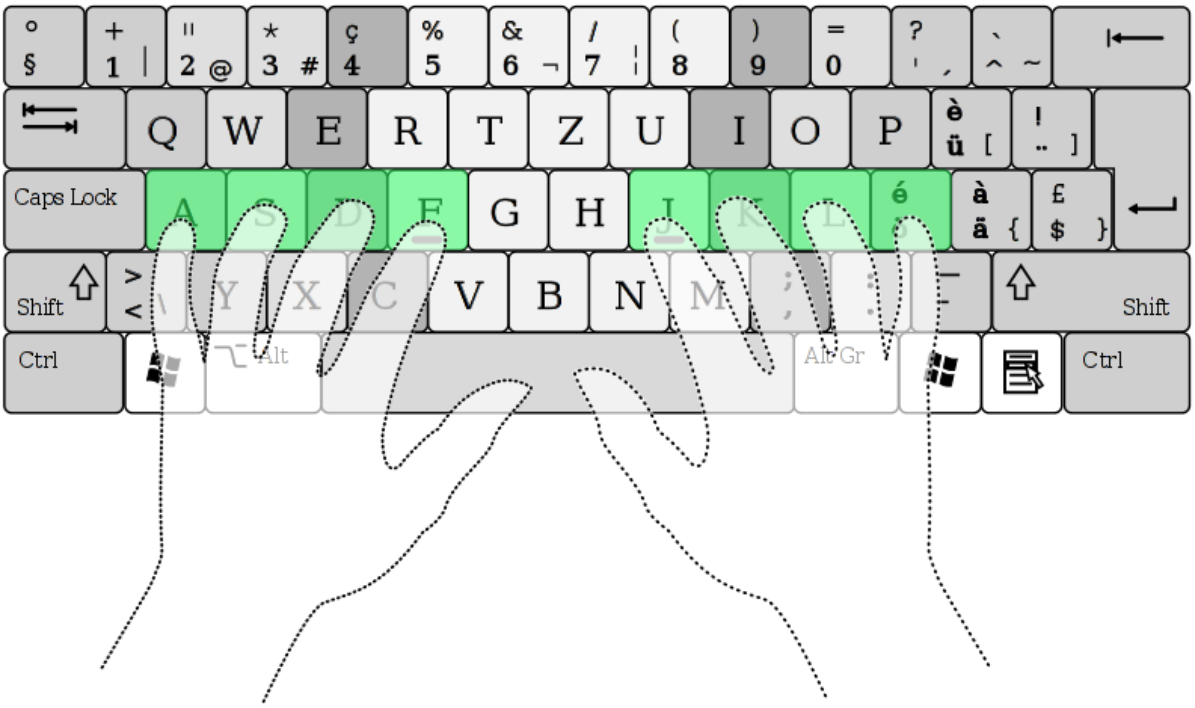


Fig. 1 – Position de repos, clavier QWERTY.
Illustration par Cy21 - CC-BY-SA-3.0 (http://www.creativecommons.org/licenses/by-sa/3.0) ou GFDL (http://www.gnu.org/copyleft/fdl.html, via Wikimedia Commons http://commons.wikimedia.org/wiki/File:Typing-home-keys-hand-position.svg

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 texte : copier/coller, macros, … (c'est le mode *normal*), tantôt à sélectionner (c'est le mode *visuel*) et tantôt à insérer du texte (c'est le mode *insertion*). 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 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* (http://www.typematrix.com/). 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 : https://vincent.jousse.org/blog/fr/comment-doubler-sa-vitesse-de-frappe-au-clavier/

3.3 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` en y plaçant des valeurs que toute personne normalement constituée souhaiterait y voir figurer.

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 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
```

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.

Pour le sauvegarder avec *Vim*, il vous suffira, après avoir appuyé sur la touche `Esc (Échap)` pour repasser en mode *Normal*, de taper `:w ~/.vimrc`. Pour sauvegarder vos prochaines modifications tapez en mode *Normal* `:w`. Pour sauvegarder et quitter `:wq ~/.vimrc`. Pour quitter `:q` et pour quitter sans sauvegarder (forcer à quitter) `:q!`.

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/v2/fr/firstconfig.

Voici à quoi devrait ressembler *Vim* *après votre première configuration*.

```
 1 " VIM Configuration - Vincent Jousse
 2 " Annule la compatibilite avec l'ancetre Vi : totalement indispensable
 3 set nocompatible
 4 " -- Affichage
 5 set title
 6 " Met 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 set scrolloff=3
13 " Affiche un minimum de 3 lignes autour du curseur
14 " (pour le scroll)
15 " -- Recherche
16 set ignorecase" Ignore la casse lors d'une recherche
17 set smartcase" Si une recherche contient une majuscule,
18 " re-active la sensibilite a la casse
19 set incsearch
20 " Surligne les resultats de recherche pendant la
21 " saisie
22 set hlsearch
23 " Surligne les resultats de recherche
24 " -- Beep
25 set visualbell" Empeche Vim de beeper
26 set noerrorbells" Empeche Vim de beeper
27 " Active le comportement 'habituel' de la touche retour en arriere
28 set backspace=indent,eol,start
29 " Cache les fichiers lors de l'ouverture d'autres fichiers
30 set hidden
~
~
~
~
~
~
~
~
~
~
~
~
                                                              30,1          All
```

FIG. 2 – *Vim* après votre première configuration.

Notez l'ajout des numéros de ligne sur la gauche.

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

3.4 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
```

Vous devriez avoir un *Vim* qui ressemble à celui de la figure ci-dessous.

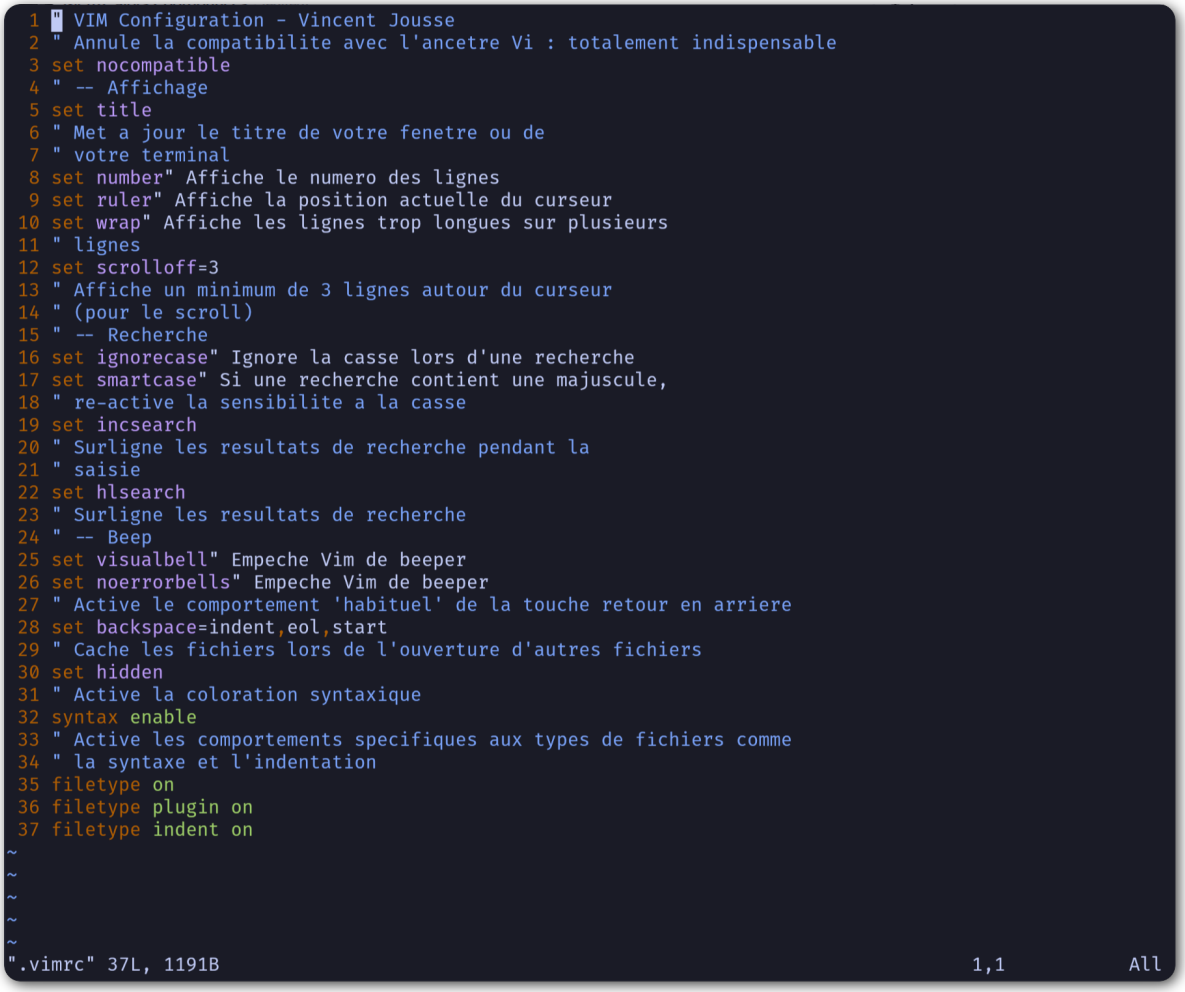


Fig. 3 – Coloration syntaxique par défaut.

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`. 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*[1] http://ethanschoonover.com/solarized (avec fond clair ou foncé, ça dépendra de vous).

https://raw.githubusercontent.com/ericbn/vim-solarized/master/colors/solarized.vim

Pour l'installer, commencez tout d'abord par créer un répertoire nommé *.vim* au même endroit que votre `~/.vimrc` (dans votre répertoire utilisateur donc). À noter que 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. Dans ce répertoire *.vim*, créez un sous-répertoire nommé *colors*. Téléchargez ensuite le fichier du thème *Solarized* https://raw.githubusercontent.com/ericbn/vim-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.

Sous Linux vous pouvez faire tout ça via les commandes suivantes :

```
mkdir -p ~/.vim/colors
wget -P ~/.vim/colors https://raw.githubusercontent.com/ericbn/vim-solarized/master/colors/solarized.vim
```

Votre répertoire *.vim* devrait ressembler à cela :

```
.vim
└── colors
    └── solarized.vim
```

Activez ensuite le thème Solarized dans votre `~/.vimrc` comme le montre le code ci-dessous. :

```
" Utilise la version sombre de Solarized
set background=dark
" Active les couleurs 24-bits dans le terminal
set termguicolors
colorscheme solarized
```

Pour tester le thème clair, remplacez *dark* par *light* (au niveau de la définition de la propriété *background*).

Ci-dessous 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 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
```

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).

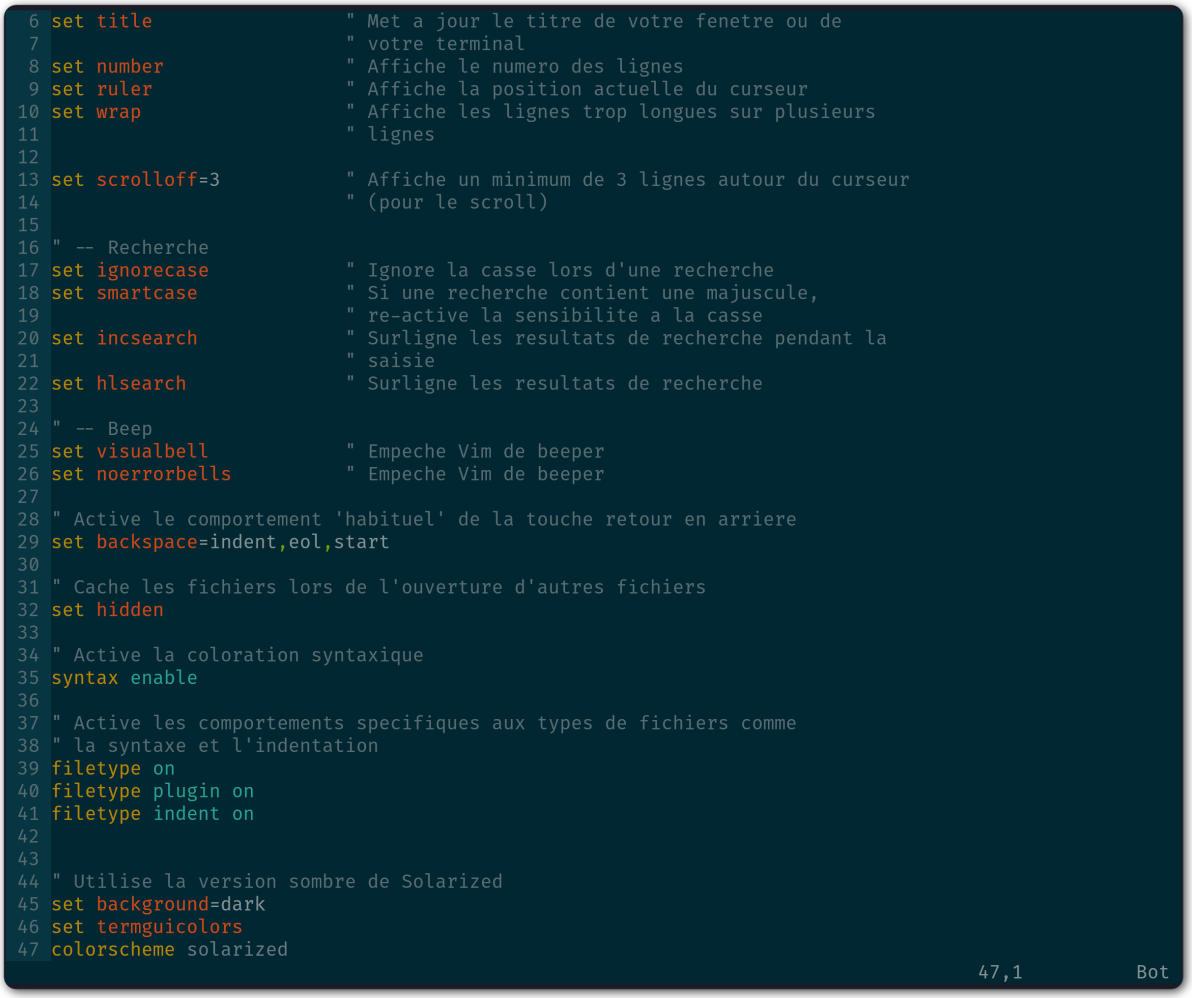
Sinon sous Linux j'utilise la police *DejaVu Sans Mono* que je trouve plutôt sympathique :

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

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/v2/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*.

1. À noter que nous utiliserons une version modernisée de *Solarized* pour vim et non l'originale disponible sur le site de l'auteur. Cette version plus récente va notamment lui permettre de fonctionner correctement sur les terminaux modernes. On l'installera à partir de ce fork https://github.com/ericbn/vim-solarized.



```
 6 set title                " Met 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 sensibilite 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 " Cache les fichiers lors de l'ouverture d'autres fichiers
32 set hidden
33
34 " Active la coloration syntaxique
35 syntax enable
36
37 " Active les comportements specifiques aux types de fichiers comme
38 " la syntaxe et l'indentation
39 filetype on
40 filetype plugin on
41 filetype indent on
42
43
44 " Utilise la version sombre de Solarized
45 set background=dark
46 set termguicolors
47 colorscheme solarized
                                                              47,1          Bot
```

FIG. 4 – Le thème *Solarized* sombre.

```
 6 set title                  " Met 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 sensibilite 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 " Cache les fichiers lors de l'ouverture d'autres fichiers
32 set hidden
33
34 " Active la coloration syntaxique
35 syntax enable
36
37 " Active les comportements specifiques aux types de fichiers comme
38 " la syntaxe et l'indentation
39 filetype on
40 filetype plugin on
41 filetype indent on
42
43
44 " Utilise la version sombre de Solarized
45 set background=light
46 set termguicolors
47 colorscheme solarized
                                                              47,1          Bot
```

FIG. 5 – Le thème *Solarized* clair.

3.5 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, ça pourrait être pratique ! Ça va être une bonne occasion pour installer notre premier plugin. 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.

3.5.1 Gestionnaire de plugins : vim-plug

vim-plug 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 *vim-plug* arrive à la rescousse, il va vous permettre d'installer chaque plugin dans un sous-répertoire rien que pour lui. Voici ce que donnerait le répertoire *.vim* d'une installation fictive de *Vim* avant et après l'utilisation de *vim-plug*.

Code source 1 – .vim avant l'utilisation de *vim-plug*

```
.vim-
├── autoload
│   └── phpcomplete.vim
├── colors
│   └── solarized.vim
└── syntax
    ├── php.vim
    └── sql.vim
```

Code source 2 – .vim après l'utilisation de *vim-plug*

```
.vim
├── autoload
│   └── plug.vim
└── plugged
    ├── solarized
    │   └── colors
    │       └── solarized.vim
    ├── php
    │   ├── autoload
    │   │   └── phpcomplete.vim
    │   ├── syntax
    │   │   └── php.vim
    │   └── autoload
    └── sql
        └── syntax
            └── sql.vim
```

Certes la version avec *vim-plug* contient plus de sous-répertoires mais chaque plugin est isolé dans son propre répertoir. Croyez-moi sur parole, ce rangement va vous éviter bien des ennuis par la suite.