ViM, LaTeX, Xfig, GNUplot et autres...

Frédéric BELLISSENT
http://ivsb2.free.fr
ivsb2@free.fr

18 avril 2017

Table des matières

Dernière version de ce document : http://ivsb2.free.fr/edition-latex.pdf

Oui, je l'affirme sans complexe,
Je suis adepte de LaTeX.

Refrain connu

Ce livret traite de la réalisation de documents LaTeX à l'aide de l'éditeur de texte Vim et de quelques autres logiciels libres.

Dans un premier temps, nous traiterons de l'utilisation de Vim pour l'édition de fichiers sources en texte brut, ce qui est la base de la réalisation de documents LaTeX.

Dans un second temps, nous verrons comment bâtir un véritable environnement de développement LaTeX autour de Vim.

Environnement de développement... pas vraiment intégré ! Ainsi, nous resterons fidèles à un des grands principes Unix qui veut que plusieurs outils combinés, dont chacun aura été conçu pour accomplir à la perfection une tâche précise, feront mieux qu'une seule application «tout intégré», plus lourde. Attention toutefois : pas intégré ne veut pas dire pas cohérent...

Nous aborderons le plus concrètement possible l'utilisation et le réglage d'applications libres pour :

— visionner les documents produits (Xdvi et Xpdf),
— dessiner en vectoriel (Xfig),
— tracer des courbes mathématiques (Gnuplot).

Il s'agit des applications libres «historiques» pour chacune de ces tâches. Elles ont largement fait leurs preuves (doux euphémisme) et on est à peu près sûr de les retrouver dans pratiquement tous les systèmes Unix.

En fait, un système X-Window minimal leur suffit, ce qui réduit d'autant les dépendances au moment de l'installation et explique leur existence dans de nombreux systèmes, y compris le très particulier Cygwin. En particulier, elles ne dépendent pas des librairies Gtk ou Qt des bureaux Gnome ou KDE. Et leur fonctionnement est tout aussi léger que leur installation.

Des applications plus «modernes» existent, avec d'autres possibilités et même si je les utilise maintenant couramment, elles ne sont clairement pas la priorité de cette documentation.

Tout au long de cette documentation, nous verrons des modèles LaTeX ou des extraits de fichiers de configuration pour Vim et les autres applications. Je laisse leurs versions complètes en libre téléchargement.

Chapitre 1

Réaliser des documents LaTeX

1.1 Comparaison, principes

Comparer le «traitement de textes» et un logiciel de mise en page à balises, comme LaTeX, est une vaste question.

L'appellation traitement de textes est d'ailleurs un peu abusive tant les interventions de l'utilisateur rendues nécessaires par ce type de logiciel peuvent s'avérer nombreuses, laborieuses... Et pour un résultat pas toujours à la hauteur.

La principale caractéristique du «traitement de texte» est de tenter de concilier simultanément les modifications du texte proprement dit, le contenu, et le rendu «final» à l'écran. Cette caractéristique fondamentale du traitement de texte est aussi, selon moi, son principal travers, car on en vient à mêler en permanence saisie, mise en page et structuration — à supposer que l'on soucie un tant soit peu de cette dernière.

Bien sûr, une utilisation avancée, voire experte, de ce type de logiciel est possible mais au prix d'une lourdeur certaine. Peu importe, d'ailleurs, qu'il s'agisse d'Abiword, de LibreOffice Writer ou de Microsoft Word : libres ou propriétaires, lourds ou légers, tous sont concernés.

Premier exemple : les listes à puces ou numérotées. Quel utilisateur de traitement de texte, même confirmé, ne s'est jamais débattu avec cette fonction pourtant très banale ?

Typiquement, on perd la numérotation en cours de liste ou, au contraire, la liste s'active sans raison sur le paragraphe suivant que l'on éditait normalement jusqu'alors. Dans les deux cas, quelques tâtonnements sont nécessaires (effacement, réécriture, annulations) pour retrouver un comportement cohérent du traitement de texte, sans forcément que l'on comprenne comment on a résolu le problème.

Avec LaTeX :

```
\begin{enumerate}
\item bla-bla
\item pouic !
\item pouet-pouet
\item Meuh !
\end{enumerate}
```

donnera :

1. bla-bla
2. pouic !
3. pouet-pouet
4. Meuh !

après compilation du fichier source LaTeX.

Ce simple extrait de code montre les deux étapes de la réalisation d'un écrit avec LaTeX :

on tape son code puis on le compile afin de produire le document sous sa forme finale, lisible ou imprimable normalement.

Il est évident que la liste peut à volonté, être coupée, réordonnée ou complétée sans jamais rendre la numérotation incohérente ni perturber les paragraphes situés avant ou après, puisque LaTeX ne met pas à jour le document «final» simultanément, mais seulement s'il est compilé *volontairement*.

Second exemple : la table des matières. Comme elle est appelée à changer au moindre ajout de contenu, il est plus rentable, lors de la rédaction d'un document structuré, même s'il n'est pas très long, d'établir une table des matières avec actualisation automatique des numérotations en chapitres, sections, pages, etc...

Voici comment le chapitre et la section où nous nous trouvons ont été écrits dans le fichier source :

```
\chapter{Réaliser des documents \LaTeX{}}

\section{Comparaison, principes}

Comparer le «traitement de textes» et un logiciel de mise en page à
balises, comme \LaTeX{}, est une vaste question.
```

Là encore, aucun risque d'incohérence si on modifie ou si on réordonne des titres ou des parties déjà écrites : un fichier source LaTeX écrit proprement n'est porteur par lui-même d'aucune numérotation fixe ; la gestion de la numérotation est entièrement à la charge du *compilateur* LaTeX, vous n'avez pas à vous en soucier.

Ensuite, LaTeX exige un effort surhumain de votre part : à l'endroit où vous souhaitez positionner la table des matières dans le texte, il faut écrire la commande `\tableofcontents` . C'est tout. Et il ne sera plus nécessaire d'y revenir.

Observons une minute de silence pour tout ceux qui se sont tapé une table des matières *manuelle* en traitement de texte. Et une autre minute de silence pour leurs lecteurs, ils la méritent...

Prout !

Ça y est.

Pour donner le même résultat, pour gérer automatiquement les numérotations, en fait, tout traitement de texte réclame également tout au long de la rédaction du document de *marquer* [1] les titres pour structurer en chapitres, sections, annexes... Bref, comme LaTeX. Puis d'insérer explicitement la table des matières à l'endroit voulu dans le texte. Bref, comme LaTeX.

Sauf que, contrairement à LaTeX, tout ceci doit être fait par sélections des titres et clics de souris successifs dans des menus plus ou moins imbriqués jusqu'à la fonction de marquage au lieu de taper `\chapter`, `\section`, etc... au sein même du texte. Et c'est beaucoup plus long ! Et c'est finalement le plus amusant, surtout pour qui a déjà écouté des anti-LaTeX ramener leur science.

1. Marquer au sens logique du terme. C'est-à-dire signaler au logiciel que tel texte est un titre de section ou de chapitre. Changer seulement la police de caractères ou la graisse ne suffit évidemment pas et serait encore plus fastidieux. Imaginons que l'on souhaite changer *a posteriori* l'aspect visuel de certains éléments, par exemple la police de tous les titres de 2e niveau sur la totalité d'un document, cela prendrait un temps tout simplement inacceptable.

Les listes à puces et la table des matières ne sont finalement que deux exemples emblématiques des nombreux éléments de mise en page dont la stabilité est loin d'être «garantie» avec un traitement de texte classique.

En fait, même réaliser une présentation élégante et homogène... Oui, bon, je sais que ce n'est pas possible... Disons simplement homogène... peut devenir un vrai calvaire avec un traitement de texte.

LaTeX, lui, mérite pleinement le nom de logiciel de mise en page : il fait automatiquement et intégralement toutes ces tâches. Vous vous chargez de la rédaction et de la structuration ; il se charge de la mise en page. Il a été conçu pour cela dès l'origine. Il ne fait que cela et il le fait bien. Nous allons voir comment en faisant d'abord la distinction entre *éditer un fichier source* et *produire un document*.

Le premier travail – le vôtre ! – consiste à écrire un fichier en texte brut portant l'extension `.tex`. Le *fichier source* contient l'essentiel, c'est-à-dire le texte de votre futur document, au format texte brut, saisi au kilomètre, avec pour seule «mise en page» un saut de ligne marquant le changement de paragraphe [2].

Tout le reste est accompli par l'écriture de balises au sein de ce même fichier source, comme dans les deux exemples précédents.

Les balises sont des commandes propres à LaTeX ; elles ne font pas partie du contenu mais précisent, le cas échéant :

— sa structure : chapitre, section, sous-section, autre,
— sa nature : table des matières, renvoi en marge ou en bas de page, références croisées, bibliographie, liste numérotée,
— son aspect : mise en gras ou en italique, alignement, etc...

Toujours par du code, LaTeX pourra composer des objets autres que du texte classique, comme des tableaux, des symboles mathématiques ou chimiques, des dessins (oui, il existe des commandes LaTeX pour dessiner !) ou encore insérer une image à partir d'un fichier externe.

Naturellement, les balises n'apparaissent pas dans la version imprimable.

Le second travail, la *production du document*, est fait à la compilation de votre fichier source par LaTeX et la «vraie» mise en page aura été réalisée sans travail supplémentaire de votre part.

Tous les éléments susceptibles de varier au moindre rajout de contenu – pagination, numérotations diverses, listes, table des matières, renvois, références croisées, bibliographie – sont réactualisés automatiquement à chaque nouvelle compilation et la nécessité de retoucher manuellement la mise en page demeure exceptionnelle.

L'essence même de la stabilité d'une mise en page LaTeX tient à cette séparation des phases de rédaction et de compilation.

LaTeX ne cherche pas la facilité à tout prix : son mode de fonctionnement, qui passe pour un manque de convivialité (vocable à la con !) auprès des intoxiqués de la souris, constitue en fait toute sa puissance. Et comme tous les outils puissants ou spécialisés, il faut apprendre à l'utiliser... Désolé.

Par contre, comme c'est un logiciel libre, chacun peut en profiter sans discrimination, la seule «mise de fonds» indispensable étant un petit effort initial de documentation et d'apprentissage. C'est l'affaire d'une heure ou deux, sans se presser, que de maîtriser la saisie de

2. Un retour à la ligne simple sera interprété comme une espace unique. Plusieurs espaces également. Tout ceci laisse une certaine souplesse pour aérer et indenter son fichier source.

textes généralistes. Pour des textes mathématiques, par exemple, il faut juste rajouter un peu de temps d'apprentissage.

Quoi qu'il en soit, LaTeX reste le plus performant pour réaliser un document à la typographie irréprochable [3] et parfaitement structuré, quelle que puisse être sa longueur.

Beaucoup de gens pensent et affirment le contraire sans complexe et surtout... sans l'avoir essayé ! Ils ont naturellement le droit de ne pas aimer LaTeX et de le dire mais seraient mieux inspirés de ne pas nier un fait incontestable : aucun des traitements de texte actuels ne parvient au niveau de puissance, de qualité et de fiabilité de LaTeX ; leur conception ne le leur permet tout simplement pas.

1.2 Formats de fichiers

Le format initial, on l'a dit, est le format texte brut, identifié par l'extension `.txt` dans certains systèmes. Pour les fichiers sources LaTeX, on utilisera plutôt l'extension `.tex` mais cela ne change rien au contenu du fichier.

D'ailleurs, dans les vrais systèmes, pas besoin d'extension...

Il sera plus important de veiller au codage de caractères. Pour le français, même s'il est possible d'utiliser le codage universel UTF-8, le codage «Latin 1» ou ISO-8859-1 reste un très bon compromis entre l'aspect pratique et la compatibilité, : il permet la saisie directe des cédilles et accents et est lisible directement par la majorité des éditeurs de texte dans tous les systèmes (BSD, Linux, Mac OS X, Windows [4]). Le codage peut être choisi ou vérifié dans tout éditeur de texte un peu evolué (pas Microsoft Notepad, donc) ou dans les préférences du système utilisé. Au besoin, on pourra convertir le fichier dans un autre codage.

La question du codage réglée, nous pourrons donc travailler notre texte même sur une machine qui ne nous appartient pas et même si les compilateurs LaTeX ne sont pas installés, la compilation étant alors remise à plus tard.

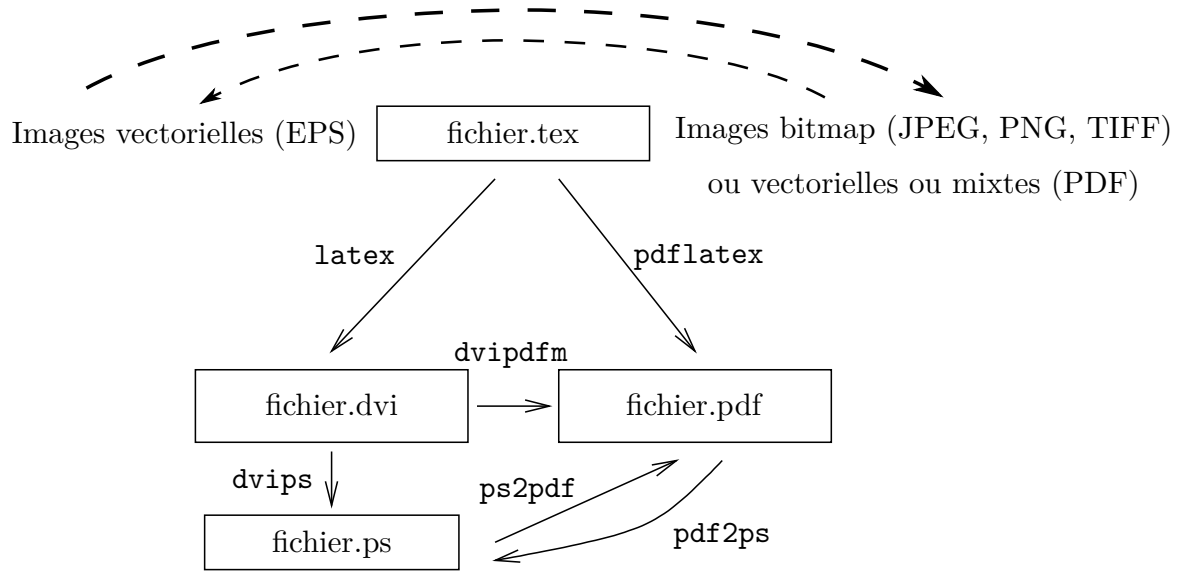
Selon le compilateur utilisé, LaTeX ou PDFLaTeX, on peut obtenir comme format final du DVI, du Postscript, du PDF. D'autres compilateurs pourront produire des pages HTML.

Format final, c'est vite dit : le format DVI (*DeVice Independent*) est très pratique pour la visualisation mais n'est finalement qu'un intermédiaire avant d'autres conversions. En particulier, si l'on veut imprimer, il faudra au moins convertir en Postscript ou en PDF.

Pour cette raison, depuis quelques temps, sauf cas particulier, je produis directement du PDF et adapte toute ma chaîne de production en conséquence, en particulier le type d'images à inclure.

3. LaTeX a été *conçu* d'après les règles de typographie. Les traitements de texte, on peut en douter...

4. Allez ! Petite écorchure supplémentaire : Windows... Vous savez, ce système qui fonctionne dans ses entrailles en UTF-8 mais qui n'offre pas en standard à son utilisateur la possibilité d'afficher correctement un texte brut en UTF-8...



Si l'on a des contraintes particulières au niveau des formats d'images ou du format final, il faut savoir qu'il existe toujours une chaîne de compilation ou de conversion permettant d'arriver à ses fins, soit en convertissant les images au préalable, et là, les outils sont innombrables, graphiques ou en ligne de commande, soit en convertissant le document *a posteriori*.

Sans parler des facilités présentes dans les systèmes Unix pour manipuler des fichiers PDF en ligne de commande : génération d'un nouveau PDF à 2 pages ou plus par feuille, constitution d'un livret PDF destiné à être rélié ou broché. Il est même possible, si l'on a écrit un document de moins d'une demi-page, de créer un PDF d'une page A4 où il figurera en double. Les possibilités sont très nombreuses et guère bridées que par notre imagination ou nos usages.

En outre, il existe de plus en plus d'interfaces graphiques à ces outils en ligne de commande.

Passons sur la stabilité et la compatibilité quasi-universelle de formats tels que Postscript ou PDF et concentrons-nous pour le moment sur l'édition de fichiers sources pour LaTeX.

Le fait qu'il faille éditer un fichier source en texte brut puis le compiler assimile plus le travail avec LaTeX à de la programmation qu'à du traitement de textes classique et pour LaTeX comme pour tout langage de programmation, deux grandes catégories d'applications peuvent être utilisées.

1.3 Les IDE

I.D.E. : *Integrated Development Environment*. Environnement Intégré de Développement. Un I.D.E. est une application unique comprenant au moins un éditeur de textes. Des fonctions supplémentaires permettent d'insérer rapidement des éléments du langage utilisé par le biais de menus texte ou d'icônes. Une aide succinte sur le langage — syntaxe ou signification — est parfois disponible.

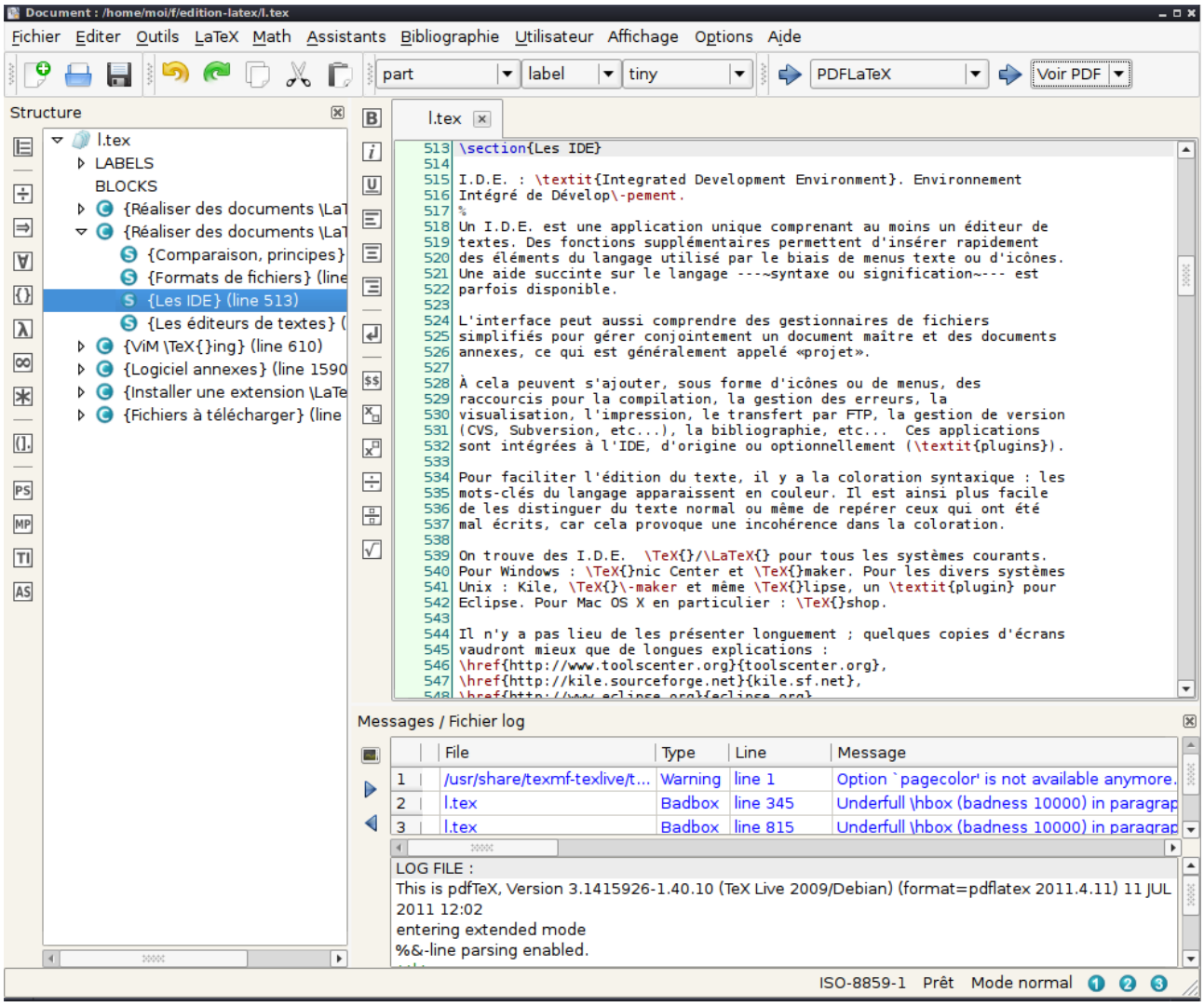
L'interface peut aussi comprendre des gestionnaires de fichiers simplifiés pour gérer conjointement un document maître et des documents annexes, ce qui est généralement appelé «projet».

À cela peuvent s'ajouter, sous forme d'icônes ou de menus, des raccourcis pour la compilation, la gestion des erreurs, la visualisation, l'impression, le transfert par FTP, la gestion de version (CVS, Subversion, etc...), la bibliographie, etc... Ces applications sont intégrées à l'IDE, d'origine ou optionnellement (*plugins*).

Pour faciliter l'édition du texte, il y a la coloration syntaxique : les mots-clés du langage apparaissent en couleur. Il est ainsi plus facile de les distinguer du texte normal ou même de repérer ceux qui ont été mal écrits, car cela provoque une incohérence dans la coloration.

On trouve des I.D.E. TEX/LATEX pour tous les systèmes courants. Pour Windows : TEXnic Center et TEXmaker. Pour les divers systèmes Unix : Kile, TEXmaker et même TEXlipse, un *plugin* pour Eclipse. Pour Mac OS X en particulier : TEXshop.

Il n'y a pas lieu de les présenter longuement ; quelques copies d'écrans vaudront mieux que de longues explications : toolscenter.org, kile.sf.net, eclipse.org, uoregon.edu/~koch/texshop/ ou, tout simplement, une recherche de copies d'écrans sur votre moteur de recherche favori.



Les I.D.E. s'adressent aussi bien au novice qu'au spécialiste de longue date. Le fait de pouvoir insérer du code d'un simple clic de souris offre le double avantage de la rapidité et de l'apprentissage progressif du langage en observant quel code est inséré quand on clique sur une icône donnée.

Autre aspect particulièrement pratique : l'écriture des environnements. Par exemple, pour centrer du texte, on clique sur l'icône appropriée[5] et les balises :

```
\begin{center}
```

5. La même que dans un traitement de texte, hi, hi !

```
\end{center}
```

sont insérées. On peut alors écrire entre elles le texte voulu : le curseur d'écriture est même déjà positionné !

Encore mieux avec certains éditeurs ou I.D.E. : si le texte à centrer est déjà écrit, on le sélectionne à la souris et on clique sur l'icône en question ; les balises seront placées immédiatement *autour* de ce texte. Même chose pour une balise en ligne : pour mettre un ou plusieurs mots en gras, les sélectionner puis cliquer sur l'icône voulue. `\textbf{` et `}` seront placés autour.

1.4 Les éditeurs de textes

L'élaboration d'un document LaTeX peut donc se résumer, on l'a vu, à la saisie d'un fichier source en texte brut. L'éditeur de texte le plus basique qui soit peut donc convenir. Pour travailler temporairement sur une machine où LaTeX n'est pas installé, cela reste un avantage énorme et même... la condition nécessaire !

Mais sur sa propre machine, où l'on est parfois appelé à travailler pendant de longues heures, autant utiliser l'éditeur avec lequel on est le plus à l'aise et que l'on pourra largement personnaliser. Et là, pas question de s'imposer l'éditeur de texte le plus spartiate (incomplet ?) possible pour le seul plaisir de travailler moins confortablement et moins rapidement. Ce serait d'ailleurs difficile après avoir goûté aux I.D.E. et à tous leurs aspects très pratiques.

L'éditeur dont je vais parler maintenant, Vim, fait preuve d'une efficacité et d'une légèreté tout simplement absentes des I.D.E., sans nier un instant leurs nombreuses autres qualités.

À qui sait le configurer, il peut offrir la plupart des caractéristiques d'un bon IDE et d'autres qui lui sont exclusives. Seule contrainte : avoir eu un bref aperçu de ses modes standard de fonctionnement, faute de quoi on continuera de croire qu'il n'est bon qu'à écrire laborieusement quelques caractères avant de quitter par `Escape :wq`.

Chapitre 2

ViM TEXing

Passons rapidement en revue les principaux avantages de Vim :

— si LaTeX est disponible pour un système d'exploitation donné, Vim le sera aussi,
— il permet de créer de nouvelles commandes en combinant à volonté ses commandes existantes,
— il permet de *choisir* des raccourcis pour toute action ou enchaînement d'actions, sans aucune exception,
— il permet d'exécuter des commandes externes,
— il permet l'enregistrement de macros,
— il est programmable dans son fonctionnement même : on peut lui rajouter des fonctions et, dans sa version graphique, des menus personnalisés, icônes comprises,
— tous les paramétrages qui seront évoqués dans cette documentation peuvent être effectués en modifiant un unique fichier de configuration [1], adaptable d'un système à l'autre en ne changeant que quelques lignes,
— c'est un logiciel libre !

Ajoutons que Vim existe sous deux formes : Vim, en mode texte, utilisable dans une fenêtre DOS, une console ou un émulateur de terminal dans le monde Unix. Et en mode graphique, nous trouvons gVim, pour Mac OS X [2], GNU/Linux et tous les systèmes Unix muni de X-Window ou encore Windows.

Dans ce qui suit, sauf précision, Vim pourra signifier indifféremment Vim ou gVim. Hormis ces deux versions «standard», il existe aussi un certain nombre de dérivés ; certains ont un fonctionnement simplifié, plus proche de celui des éditeurs conventionnels. Cream, par exemple, n'a pas, au démarrage, le fonctionnement *modal* de Vim : voir cream.sourceforge.net .

Venons-en, justement, à ce fonctionnement *modal* plutôt déroutant lors des premières utilisations de Vim. Il y a quatre modes de fonctionnement principaux :

mode normal : les touches du clavier ne servent pas à saisir du texte mais à commander Vim : déplacements, manipulations simples sur du texte déjà écrit,

mode commande : accès à d'autres fonctions comme sauvegarde et ouverture de fichiers, recherche, transformations de texte plus complexes, exécution de commandes externes sans quitter Vim, réglages de Vim, etc...

mode édition : saisie normale de texte,

1. Je ne prétends pas que ce soit la méthode la plus puissante ou élégante...
2. Voir aussi le récent projet MacVim, qui s'adapte encore plus étroitement à l'interface graphique de ce système : `http://macvim.org`.

mode visuel : sélection de texte, au clavier ou à la souris.

2.1 Options générales

Comme toutes les autres options, elles sont enregistrables telles quelles, pour un utilisateur individuel ou pour tous les utilisateurs d'un système, dans un simple fichier au format texte brut. Plus loin, nous verrons où et comment les enregistrer.

- `set tw=72` : largeur de texte bien adaptée à l'inclusion de code LaTeX dans des e-mails et à l'édition en Xterm, même si on active la numérotation des lignes,
- `set ts=4` : inutile que les tabulations fassent 8 caractères...
- `set vb` : *visual bell* ; remplace les bips d'erreur par un flash de l'affichage,
- `syntax on` : coloration syntaxique active,
- `set ai` : indentation automatique,
- `set nohls` : pas de coloration des résultats de recherche,
- `set number` : activer la numérotation des lignes,
- `set ignorecase` : recherche insensible aux majuscules/minuscules...
- `set smartcase` : ... sauf si on tape certaines lettres en majuscules,
- `set mouse=a` : étend les possibilités de la souris[3],
- `set noguipty` : influence la façon dont gVim transmet les commandes externes au système. Attention : cela peut modifier l'affichage de certaines commandes console, même simples, comme `ls`. Option réservée à gVim, la version graphique.

Tous ces réglages peuvent être changés à la volée en mode commande[4] en tapant par exemple `:set ts=72`, `:set hls`, `:set nohls`, `:set nu`, `:set nonu`, etc... Sans les virgules, naturellement ! Pour obtenir de l'aide, par exemple sur `ts` (ou `tabstop`), taper, en mode normal, `:help tabstop` ou `:he ts`

Pour avoir des détails sur le dernier réglage de la liste, notamment, vous pouvez taper `:h gui-pty`

Attardons-nous un peu sur la coloration syntaxique que l'on active par `:syntax on` et désactive par `:syntax off`, indépendemment du type de fichier. Si la coloration standard ne vous convient pas, on peut en trouver d'autres sur Internet : voir vim.org pour plus de détails.

Les schémas de coloration sont des fichiers de configuration particuliers (de simples fichiers texte, eux aussi). Il suffit d'enregistrer le schéma trouvé (par exemple `kate.vim`) dans le dossier caché `~/.vim/colors` de votre dossier personnel. L'administrateur fera de même dans `/usr/share/vim/vimcurrent/colors` s'il veut le diffuser à tous les utilisateurs.

On peut changer de schéma en tapant en mode commande `:colo bidule` (colo comme `colorscheme`). Vous constaterez avec plaisir que le complètement des noms de schémas est possible avec la touche TAB, exactement comme les commandes et noms de fichiers dans le shell Bash. Avec gVim, on utilise un menu déroulant[5].

Je n'ai jamais pris Vim en défaut dans le domaine de la coloration syntaxique. Clarté et absence de bugs sont au rendez-vous. Le schéma standard (`default`) m'a toujours convenu, tant pour Vim dans le Xterm à fond noir de Debian que pour gVim. J'ai même reproduit pour gVim ce fameux schéma «Xterm-Debian» : xterm-debian.vim sur ma page.

3. Positionnement du curseur, sélection et copier-coller, essentiellement. Permet une édition plus «moderne». Personne n'a dit plus efficace...
4. Assurez-vous au préalable d'être revenu en mode normal à l'aide de la touche *Esc.*
5. Au fait : avez-vous vu que ces menus étaient détachables ?

Ceci est essentiellement affaire de goût ; je n'ai donc pas consacré beaucoup de temps à cela.

Après, selon les réglages du serveur X ou les couleurs réglées pour XTerm, il m'a fallu, une fois ou l'autre, régler dans mon `vimrc` l'option :

```
set background=light
```

Il est évident que pour d'autres schémas de coloration, il faudra sans doute l'option contraire.

2.2 Enregistrer des réglages pour Vim

Avec la Debian, un premier fichier possible est `/etc/vim/vimrc`, c'est-à-dire le fichier commun à tous les utilisateurs. Naturellement, il faut les droits d'administration et cet emplacement varie selon le système utilisé ; dans un système Unix, la commande :

```
$ find / -name "*vimrc*"
```

vous aidera à le trouver. À en trouver plusieurs, très certainement. Ne pas taper le caractère `$`, il signale simplement que cette commande peut être lancée à partir d'un compte utilisateur normal.

Chaque utilisateur peut aussi écrire dans `~/.vimrc`. Il s'agit du `vimrc` caché du dossier personnel [6]. C'est la bonne solution si l'on n'a pas assez de droits sur le `vimrc` commun ou si l'on veut des réglages personnels, supplémentaires ou temporaires. Si on les change fréquemment, aussi.

Quoiqu'il en soit, puisqu'il s'agit de personnalisation, autant faire le minimum au niveau système, dans le dossier `/etc`, et laisser à chacun le soin de personnaliser son `vimrc`, dans son dossier personnel.

2.2.1 Intégrer au mieux de nouveaux réglages

Le plus simple *semblerait* donc d'enregistrer tout nouveau réglage dans un `vimrc` préexistant ou même de trouver un `vimrc` complet pour remplacer le sien. En fait, je déconseille désormais ces deux démarches.

Contrairement à ce que je faisais auparavant, je ne fournis plus de `vimrc` complet mais un fichier dans le même langage, ivsb2.vimrc, à utiliser *en complément* de votre `vimrc` existant. Ainsi, vos options initiales, peut-être indispensables au bon fonctionnement de Vim dans votre système ne seront pas perdues.

Et si vous créez votre propre `vimrc`, cette vision modulaire est également judicieuse. Des réglages complexes ou erratiques sont toujours plus faciles à annuler s'ils sont dans un fichier isolé et clairement identifié.

Si l'on pratique plusieurs langages de programmation, cette démarche est très avantageuse.

Et même, pourquoi ne pas travailler avec plusieurs `*.vimrc` reliés au `.vimrc` principal ?

6. Pour tout utilisateur, le tilde indique son propre dossier personnel ; le point indique un fichier caché.

2.2.2 Télécharger et utiliser mes réglages

Ils sont disponibles ici, sous forme d'un unique fichier : ivsb2.vimrc, adaptez-les si nécessaire et vérifiez qu'ils ne sont pas contradictoires avec les réglages que vous pouviez avoir auparavant.

Si l'on dispose des droits de Root, on les enregistre par exemple sous `/etc/vim/ivsb2.vimrc` puis on indique dans le (g)`vimrc` système :

```
if filereadable("/etc/vim/ivsb2.vimrc")
  source /etc/vim/ivsb2.vimrc
endif
```

Si l'on ne dispose pas des droits de Root, on peut enregistrer ivsb2.vimrc à la racine de son dossier personnel dans `.ivsb2.vimrc` (le point en fera un fichier caché) et écrire dans `~/.vimrc` :

```
source ~/.ivsb2.vimrc
```

Le test avec `filereadable` m'apparait moins utile dans le cadre d'une config personnelle : en général, on sait où l'on vient de déposer un unique fichier ! De toute façon, en cas de problème, Vim affichera un message d'alerte.

Particularité de la version graphique, gVim : le fichier de configuration principal se nomme `gvimrc` ou `.gvimrc` au lieu de `vimrc` ou `.vimrc`, selon que l'on se trouve dans un dossier système comme `/etc` ou `/usr/share/vim` ou bien dans un dossier personnel. Les manipulations décrites précédemment restent valables, l'inclusion se fera avec la commande `source`.

Après `source`, on peut indiquer un lien symbolique mais un chemin complet est préférable.

2.2.3 Recharger son vimrc

Se fait par la commande Vim :

```
:so $MYVIMRC | e
```

`so` n'est qu'un raccourci de la commande `source`, le caractère `|` (*pipe*, habituellement en console) est un simple séparateur pour les commandes Vim. `e`, ou `edit`, permet de recharger le fichier actuel.

Pour la version graphique, évidemment, ce sera plutôt la variable `$MYGVIMRC` .

2.2.4 Et sinon...

Hors de ces deux points de vue (utilisateur individuel ou bien administrateur mettant en place une configuration multi-utilisateurs), il existe bien des façons d'enregistrer des réglages pour Vim, notamment en utilisant les dossiers `after/syntax`.

La méthode proposée ici, si elle n'est pas la plus brillante, présente au moins les avantages suivants :

— centraliser tous les réglages dans un seul fichier texte,

— et donc les rendre faciles à diffuser
— permettre une personnalisation très poussée,
— rester d'une compréhension aisée.

Et les approches évoquées ne sont pas exclusives : rien n'empêche de conserver certains réglages au niveau du `vimrc` système et d'autres au niveau du `vimrc` personnel, plus adapté à des modifications personnelles ou fréquentes, ni d'avoir d'autres réglages répartis dans les dossiers `after/syntax`.

2.2.5 Dernières précisions

On peut adapter le fichier ivsb2.vimrc d'un système à l'autre en ne changeant que quelques lignes, notamment les chemins vers les modèles de fichiers LaTeX ou vers certains exécutables, pour les systèmes aux *paths* un peu tordus, suivez mon regard vers les fenêtres...

Important aussi : ce fichier peut être utilisé indifféremment avec les deux versions de Vim, texte et graphique. Les fonctions éventuellement incompatibles entre ces deux versions de Vim, sont exécutées seulement après tests au sein du fichier `ivsb2.vimrc`. Seul impératif : ne pas oublier de faire une inclusion dudit fichier dans `vimrc` *et* dans `gvimrc`.

Personnellement, j'utilise couramment Vim et gVim, selon le travail à faire, et ne peux donc dire *a priori* lequel est le plus efficace.

Avec des versions récentes de Windows, par exemple, gVim est tout de même plus utilisable et ne dépend pas de la stabilité aléatoire ni des limitations techniques de la fenêtre MS-DOS rebaptisée «invite de commande» sous NT/XP pour faire croire que ce n'est pas pareil.

Avec Unix, système où la console est beaucoup plus performante et présente, c'est plus nuancé. gVim peut s'avérer plus pratique si l'on utilise souvent un gestionnaire de fichiers graphique mais la version console, Vim, plus proche du shell, offre une grande puissance et des particularités à ne pas négliger.

Et puis, pour éditer du texte... Des outils en mode texte ne sont pas totalement déplacés, me semble-t-il...

2.2.6 Mise en garde pour gVim !

À l'ouverture d'un fichier donné avec gVim, tous les gestionnaires de fichiers n'ont pas le même comportement !

Certains indiqueront à Vim le dossier du fichier cliqué comme dossier de travail, d'autres, non et les compilations LaTeX seront donc dirigées vers un mauvais dossier.

Message d'erreur typique : LaTeX indique qu'il ne trouve pas des fichiers d'images à inclure. Autre symptôme : on ne retrouve pas dans son fichier DVI ou PDF les modifications faites à l'instant dans le fichier source et ledit DVI ou PDF est créé dans un autre dossier.

Nautilus, gestionnaire de fichiers du bureau Gnome, fixe systématiquement le dossier personnel (`/home/utilisateur`) comme dossier de travail, même si ce n'est pas le cas... Nous sommes dans le cas typique où la console et Vim en mode texte s'avèrent plus efficaces, moins trompeurs, du moins, avec les réglages standard.

Thunar, lui, ne présente pas un tel problème.

Heureusement, des solutions existent. La commande Vim `:pwd` (*Print Working Directory*)

permet de vérifier le dossier de travail actuel et `:lcd` (*Change Directory*) permet d'en changer.

L'aide de Vim (`:help :cd`), vous apprendra même que `:cd %:h` fixe le dossier du fichier édité comme dossier de travail. Voyez également `:help :lcd`. Mon fichier ivsb2.vimrc comprend une commande pour accomplir automatiquement cette action dès que l'on commence à éditer un fichier, quel qu'il soit ; cela tient en une courte ligne :

```
au BufEnter * :lcd %:p:h
```

C'est automatique et, si l'on édite plusieurs fichiers (commande `:sp`), cela s'applique à chaque fichier de la fenêtre Vim.

Vim et gVim, à partir de la version 7, offrent l'option `set autochdir` ou `set acd`, en version courte. Elle change automatiquement de dossier de travail à l'ouverture d'un fichier, même en édition multiple. La ligne précédente, en commentaire dans ivsb2.vimrc, n'est conservée que pour compatibilité avec d'anciennes versions de Vim.

On peut aussi réserver ce changement de dossier à la version graphique de gVim par le test `gui_running` :

```
if has("gui_running")
    set acd
endif
```

De façon générale, dans Vim comme dans toute application, il vaut mieux savoir dans quel dossier on travaille et être attentif à toute utilisation de la commande «Enregistrer sous» (`:sav nom-de-fichier` si on travaille en mode texte) faute de quoi on risque d'enregistrer ou de compiler ses fichiers dans un dossier auquel on n'aura pas prêté attention.

2.3 Raccourcis

L'essentiel de mes personnalisations suit un double objectif : gagner en rapidité lors de la saisie et ne pas retaper toujours les mêmes commandes, plus ou moins longues, pour effectuer toujours les mêmes actions. C'est donc dans le domaine des raccourcis que le plus gros travail a été fait.

Toute suite d'actions ou de commandes accomplie dans Vim, changements éventuels de mode compris, peut être déclenchée par un raccourci. Ce que j'appelle raccourci peut être indifféremment une *combinaison de touches* (par exemple *Ctrl s*) comme il en existe dans toutes les applications et que l'on appelle habituellement *raccourci-clavier*, ou une *séquence* de touches (par exemple `,fr`).

Mes raccourcis, justement, à cause de leur nombre important, sont plutôt des séquences de touches ; le plus souvent, une virgule immédiatement suivie de lettres non-accentuées, ce qui n'arrive pas dans un texte en français. Évidemment, aucune ne comporte d'espace ou de caractère spécial.

Quant aux options de certaines balises LaTeX, comme dans :

```
\usepackage[dvips, dvipdfm]{graphicx}
```

aucun problème : les virgules de séparation peuvent être suivies d'une espace. Ainsi, pas de risque de comportement surprenant à l'édition de ce genre de code.

En outre, ce genre de séquences, nous en verrons plus loin quelques exemples, peuvent être très explicites (`,fr` pour une fraction, `,rc` pour une racine carrée, etc...).

Mais leur principal avantage est qu'elles seront toujours comprises correctement et indépendemment du système d'exploitation ou du type de clavier utilisé[7], ce qui ne serait pas le cas de toutes les combinaisons de touches basées sur *Ctrl*, *Alt* ou *AltGr* qui ont en plus le défaut ultime et définitif d'être limitées en nombre.

Pour ce qui est de l'utilisation quotidienne, précisons que Vim, avec ses réglages initiaux, arrête toute interprétation de raccourci après un certain délai. S'il est dépassé, après la frappe du premier caractère d'une séquence, le curseur se repositionne et on peut alors reprendre une frappe normale. Autrement dit, si l'on attend trop (ou suffisamment ?), une séquence n'est pas interprétée mais... écrite !

Au lecteur de faire une adaptation raisonnée de ce qui est montré ici. Par exemple, si les virgules des séquences sont incompatibles avec son usage habituel de Vim, il peut les remplacer par des *underscores* ou tout caractère de son choix.

2.3.1 Raccourcis permanents

Comme les options ci-dessus, ils ne dépendent pas du type de fichier.

Sauvegarde du fichier actuel	`F2`
Ouverture d'un terminal (Xterm) dans le dossier de travail actuel	`Shift F2`
Formatage de la sélection ou jusqu'à la fin du paragraphe actuel	`F6`
Établir le dossier du fichier actuel comme dossier de travail	`,dt`

2.3.2 Raccourcis LaTeX : compilation, appel à des programmes annexes

Xfig	`F3`
Kcalc	`Shift F3`
Sauvegarde et compilation	`F4`
Visualisation DVI	`F5`
Compilation directe en PDF	`Shift F4`
Visualisation PDF	`Shift F5`
Conversion DVI en PDF	`F9`
Conversion Postscript	`F7`
Visualisation Postscript	`F8`
Demi-format (Postscript)	`Shift F7`
Visualisation demi-format	`Shift F8`
Insertion d'un fichier modèle	`,mod ou ,ex`

7. Penser aux claviers des stations Sun, HP ou SGI, des Mac, etc... ou tout autre n'ayant pas la même disposition de touches.

2.3.3 Balises LaTeX : structure

Section	`,s`
Sous-section	`,ss`
Sous-sous-section	`,sss`
Section (non-numérotée)	`,s*`
Sous-section (non-numérotée)	`,ss*`
Sous-sous-section (non-numérotée)	`,sss*`

2.3.4 Balises LaTeX : mise en forme

Saut de ligne	`,lb`
Saut de ligne (sans justification)	`,cr`
Centrage	`,c`
Aligné à gauche	`,l`
Aligné à droite	`,r`
Gras	`,bf`
Italique	`,i`
Petites capitales	`,sc`
Caractères machine à écrire	`,tt`
Verbatim (en ligne)	`,vb+`
Verbatim (environnement)	`,vbt`
Fantôme	`,ph`
Insertion tableau	`,tab`
Liste	`,it`
Énumération	`,en`
Élément de liste ou d'énumération	`,ite`

2.3.5 Balises LaTeX : mathématiques

Fraction	`,fr`
Taille math standard	`,dst`
Insertion d'une figure	`,igr`
Insertion d'une figure en paragraphe	`,ppl ou ,ppr`
Racine carrée	`,rc`
Signe multiplié	`,*`
Vecteur	`,v`

2.4 Comprendre les raccourcis

Tout d'abord, je précise que je n'ai montré que quelques raccourcis à titre d'exemple, en particulier pour la partie «**Mathématiques**» où ils sont pourtant les plus nombreux.

Naturellement, avec sélection préalable (mode visuel ou souris), toutes les balises englobantes peuvent être appliquées à du texte déjà écrit.

Pour bien comprendre les raccourcis ou en élaborer soi-même, il est nécessaire de connaitre quelques commandes Vim de base. Voici celles qui m'ont servi un jour ou l'autre : vous en retrouverez une grande partie dans ivsb2.vimrc.

Attention aux majuscules/minuscules : en cas d'erreur, vous risquez de faire le contraire de ce que vous souhaitez !

Mode normal

Mode *normal* : le clavier ne sert pas à saisir du texte mais à commander l'éditeur.

`h`	déplacement vers la gauche
`j`	déplacement vers le bas
`k`	déplacement vers le haut
`l`	déplacement vers la droite
`gg`	aller au début du fichier
`G`	aller à la fin du fichier
`37G`	aller à la ligne 37
`(`	aller en début de phrase
`)`	aller en fin de phrase
`{`	aller en début de paragraphe
`}`	aller en fin de paragraphe
`dd`	efface la ligne où se trouve le curseur
`3dd`	efface trois lignes vers le bas, dont la ligne actuelle
`p`	colle le contenu du «presse-papier» après le caractère ou la ligne actuel
`P`	colle le contenu du «presse-papier» avant le caractère ou la ligne actuel
`s`	remplace le caractère actuel.
`S`	remplace la ligne actuelle
`:`	mode commande
`i`	mode insertion (avant le caractère actuel)
`I`	mode insertion (en début de ligne)
`a`	mode insertion (après le caractère actuel)
`A`	mode insertion (en fin de ligne)
`o`	mode insertion (ligne suivante)
`O`	mode insertion (ligne précédente)
`v`	mode visuel (sélection de texte).

Quelques précisions :

— dès que quelquechose est effacé, Vim le place dans le «presse-papier» [8],
— ce que Vim appelle *remplacement* est un effacement immédiatement suivi d'un passage en mode insertion.
— de plus, à toute commande de déplacement ou à toute sélection, on peut associer copie ou effacement.

Mode insertion

On peut saisir du texte normalement [9].

`Escape`	mode normal
`Escape:`	mode commande

Mode visuel

Mode de sélection de texte. On y accède depuis le mode normal en pressant `v` ; on fait sa sélection en effectuant un déplacement quelconque à l'aide du clavier. On peut aussi utiliser directement la souris si le fichier de configuration contient `set mouse=a`.

Avantage de Vim : cette sélection, même effectuée à la souris, peut être finement ajustée au clavier seul, par un déplacement quelconque (flèches, page suivante/précédente, touches début/fin).

`y`	copie la sélection dans le «presse-papier»
`d`	efface la sélection et la copie dans le «presse-papier»
`c` ou `s`	remplace la selection (l'efface puis passe tout de suite en mode insertion)
`S`	remplace les lignes contenant la sélection
`Escape`	retour au mode normal ou insertion

Toute sélection peut être copiée ou effacée strictement (`y` ou `d`, respectivement) ; on peut aussi effacer les lignes qui la contiennent (`Y` ou `D`).

Mode commande

On y accède depuis le mode normal en pressant `:` (deux points).

`:r machin.tex`	insertion du fichier `machin.tex` à la ligne suivante.
`:37r machin.tex`	insertion du fichier `machin.tex` à la ligne 38.
`:!commande_externe_options_comprises`	exécute n'importe quelle commande et affiche le résultat éventuel ; suite de l'affichage par pression sur Espace.
`:r!commande_externe_options_comprises`	exécute n'importe quelle commande et insère le résultat éventuel dans le fichier actuel, à la ligne suivante.

8. Il y a 10 presse-papier, contenant les derniers effacements ; il sont numérotés de 0 à 9.
9. Qui a dit «Enfin !» ? Ah, c'est malin !

Dans les deux derniers cas, certaines différences sont possibles par rapport à l'affichage renvoyé en console par la même commande.

Comme le shell Bash, (g)Vim permet le complètement des commandes et des noms de fichiers par deux pressions sur la touche TAB.

Des raccourcis multi-modes

On crée un raccourci en indiquant trois choses dans l'ordre : dans quel mode doit s'exercer le raccourci, la séquence du raccourci puis la frappe exacte que l'on aurait effectuée. Rigoureusement toutes les touches utilisées dans la séquence doivent figurer, y compris celles faisant changer de mode, le cas échéant.

Deux exemples suffiront à comprendre :

```
imap ,s \section{}<ESC>i
vmap ,s c\section{<ESC>pa}
```

signifie que le raccourci pour la balise de section sera `,s`. Il sera utilisable en mode insertion (écriture de la balise vide puis du texte à l'intérieur) et en mode visuel (rajout de la balise autour d'un texte préalablement écrit et sélectionné).

`imap` concerne le mode insertion. Nous écrivons `\section{}`, revenons en mode normal par Escape (`<ESC>`), ce qui nous ramène sur la dernière accolade de la balise, puis revenons en mode insertion *avant* avant cette dernière accolade (`i`). On peut alors écrire normalement le titre de section.

`vmap` concerne le mode visuel. Nous venons d'écrire puis de sélectionner à la souris ou au clavier notre futur titre de section sans balise. Nous remplaçons la sélection par la balise `\section{` jusqu'à la première accolade. Nous quittons alors le mode insertion (`<ESC>`), ce qui nous ramène sur la première accolade. Nous collons *après* elle (`p`) la sélection précédemment effacée ; le curseur est alors sur le dernier caractère de cette sélection. C'est le moment d'insérer la dernière accolade *après* lui : on revient en mode insertion *après le curseur* (`a`) puis on tape `}`.

```
imap ,c \begin{center}<CR>\end{center}<ESC>O
vmap ,c S\begin{center}<CR>\end{center}<ESC>P
```

Ici, le raccourci `,c` doit permettre, en mode insertion, l'écriture des balises de centrage en laissant une ligne vide entre elles et, en mode visuel, l'écriture des balises de centrage autour de lignes que l'on aura sélectionnées.

En mode insertion (`imap`), nous écrivons la première balise, sautons à la ligne par Entrée (`<CR>`), écrivons la seconde, retournons en mode normal (`<ESC>`) et enfin, en mode insertion *entre les deux lignes précédentes* (`O`).

En mode visuel (`vmap`), nous sélectionnons le texte à centrer, effaçons *les lignes qui le contiennent* (`S`), revenons en mode insertion (`i`) pour écrire la première balise, sautons une ligne par Entrée (`<CR>`), écrivons la seconde balise, quittons à nouveau le mode insertion (`<ESC>`) et enfin, collons *les lignes effacées* entre les deux balises (`P`).

Toute autre balise englobante, sur une ou plusieurs lignes, suit le même principe. Prenez le temps de bien comprendre ces raccourcis avant de créer les vôtres. Copiez et adaptez. Le seul fait de changer le raccourci lui-même (mais pas l'action effectuée) pour le rendre plus explicite n'est déjà pas si mal...

2.5 Usage de fonctions

Parmi les innombrables caractéristiques de Vim, il est naturel de trouver la reconnaissance du type de fichier. Type texte évidemment, mais la lecture de l'extension permettra à Vim, par exemple, d'appliquer correctement la coloration syntaxique, une indentation adaptée, ou que sais-je...

Et on est loin d'être limité à cela.

En lisant le fichier de configuration proposé, vous constaterez qu'il permet un traitement différencié des fichiers TEX et HTML. Les lignes suivantes :

```
au BufEnter,BufNewFile *.tex        exe Flatex()
au BufEnter,BufNewFile *.html,*.htm,*.php       exe Fhtml()
```

vous permettront d'exécuter la fonction `Flatex` ou `Fhtml` selon la nature du fichier à éditer, à condition que les fichiers aient l'extension `tex` attendue. Si le fichier n'existe pas encore, il faut lancer Vim avec la commande :

```
vim nom-de-fichier.extension
```

afin d'activer tout de suite les coloration et fonction adaptées.

Les fonctions en question peuvent être copiées purement et simplement, même dans des systèmes autres que Linux [10].

On peut éviter certaines lacunes de coloration avec les fichiers nouvellement créés en saisissant :

```
au BufEnter,BufNewFile *.tex setfiletype tex
```

ou encore :

```
let g:tex_flavor = "latex"
```

qui impose de colorer pour le langage LATEX et non TEX.

Dernière astuce : pour éviter certains messages d'erreur à la lecture des fonctions, les déclarer par `function!` (point d'exclamation). Ce type de message arriverait par exemple au rechargement des `vimrc`.

2.6 Des variables dans `vimrc`

En parcourant le `vimrc`, on pourra voir pas mal de commandes qui font appel au même exécutable mais parfois dispersées dans le fichier. Pour plus d'efficacité, voici comment fixer pour tout le fichier, vers le début de la fonction `Flatex` la commande du visionneur PDF. Ainsi, une seule ligne à changer si tel programme ne convient plus pour telle raison.

10. Ne pas oublier, le cas échéant, d'adapter les chemins de certaines commandes.

```
let $PDFVIEWER = "xpdf"

...


" Visualisation PDF
map <F5> :! $PDFVIEWER %<.pdf &<CR>
imap <F5> <ESC>:! $PDFVIEWER %<.pdf &<CR>
```

Toutes les définitions de commandes ou de raccourcis contiennent l'appel à la variable `$PDFVIEWER` et, idéalement, le vrai nom ou chemin de l'exécutable ne doit apparaitre qu'une fois dans le fichier `vimrc`.

Inutile de dire qu'il y a aussi `$LATEXCOMPILER`, `$VIEWER`, etc...

2.7 Le problème des raccourcis de commandes

Commandes externes ou internes, d'ailleurs.

Dans le cas d'une commande externe que l'on souhaiterait lancer souvent, les raccourcis créés avec `map` ne sont pas l'idéal : d'abord, ils imposent un délai au-delà duquel la frappe clavier n'est plus interprétée mais écrite dans le fichier actuel.

Deuxième problème : même si on se restreint au mode commande seul, avec `cmap`, il se trouve que le raccourci choisi risque d'être en conflit avec un motif de recherche identique. On ne chercherait plus le motif en question mais... la commande appelée par le raccourci correspondant.

Je m'explique. Si on a créé avec `cmap` le raccourci `,sdd` qui appelle la commande externe `/home/moi/.scripts/sensdessudessous.sh`, comme ceci :

```
cmap ,sdd !/home/moi/.scripts/sensdessudessous.sh %<.pdf
```

une recherche de `,sdd` se transformerait aussitôt en une recherche de `!/home/moi/.scripts/sensdessudessous.sh`.

Les commandes internes sont alors plus adaptées. Et on créera la commande utilisateur ainsi :

```
command!  SDD !/home/moi/.scripts/sensdessudessous.sh %<.pdf
```

Remarques ou avantages :

- un nom de commande utilisateur doit commencer par une majuscule
- on préfèrera `command!` avec point d'exclamation pour éviter un message d'alerte[11] lors d'un rechargement éventuel des `vimrc`.
- plus de délai de frappe comme avec les `*map`.

11. Un pour chaque `command!`, même! Ultra-pénible. Même problème qu'avec `function!`, déjà évoqué quelques paragraphes avant.

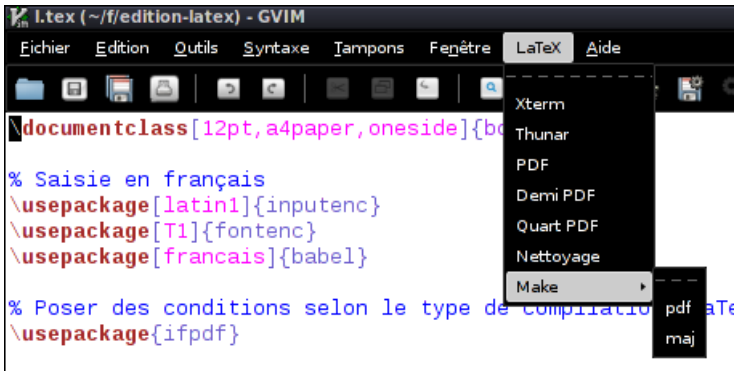
2.8 Menus personnalisés

Les lignes suivantes sont placées au sein de la fonction `Flatex`, afin de ne concerner que les fichiers `.tex`, qui plus est dans une boucle qui vérifie que l'on utilise bien la version graphique de Vim.

```
if has("gui_running")
...
    " Menu spécial LaTeX
    amenu 137.7 LaTeX.Xterm <ESC>:!xterm &<CR><CR>
    amenu 137.8 LaTeX.Thunar <ESC>:!thunar &<CR><CR>
    amenu 137.9 LaTeX.PDF <ESC>:w<CR>:!pdflatex %<.tex<CR><CR>
    amenu 137.10 LaTeX.Demi\ PDF :!pdfnup --outfile %<-miformat.pdf %<.pdf<CR>
    amenu 137.20 LaTeX.Quart\ PDF :!pdfnup --nup 4 --outfile %<-quarto.pdf %<.pdf<CR>
    amenu 137.100 LaTeX.Nettoyage :!rm -f %<.aux %<.dvi %<.log %<.out %<.ps %<.toc %<-miformat.ps <CR>
    amenu 137.100.10 LaTeX.Make.pdf :!make pdf <CR>
    amenu 137.100.20 LaTeX.Make.maj :!make maj <CR>


...
endif
```

Et... c'est tout concernant les menus personnalisés, j'y ai simplement mis les seules choses que je trouvais moins pratique avec des raccourcis. Voici le résultat :



Comme on le voit, une espace dans un intitulé de menu doit être protégée par un antislash.

2.9 Vi ou Vim ?

La plupart des possibilités évoquées depuis le début exigent Vim et non un quelconque clone du Vi originel. En console, il faut donc taper systématiquement la commande `vim`.

Avec gVim, le problème ne se pose pas.

2.10 Conclusion

Nous avons eu un bref aperçu des principales caractéristiques de Vim : présence sur la majorité des systèmes, nombreuses fonctions et commandes de base, efficacité, interaction avec le système et les autres applications, etc...

Ses possibilités de personnalisation pratiquement infinies font de Vim un éditeur à part : on peut régler avec la même finesse son interface et son fonctionnement.

Avantage définitif de Vim en ce qui me concerne : toute action, même complexe, peut être accomplie par une courte séquence de touches et le choix de telles séquences est illimité pour l'utilisateur.

Et si on a des tendances un peu plus graphicophiles... On peut se créer ses propres menus, avec uniquement le contenu et l'organisation que l'on souhaite !

On évite ainsi deux défauts fondamentaux des applications aux menus graphiques hiérarchisés :

— un temps d'accès inégal selon la fonction à utiliser,
— la limitation stricte aux choix des développeurs de ces applications (nature des fonctions et contexte d'utilisation).

On pourra naturellement objecter que travailler avec Vim force à retenir ou à régler soi-même beaucoup de choses, mais c'est toujours plus facile quand il s'agit de choix *personnels* faits dans le seul souci de l'efficacité.

Vim est effectivement moins abordable aux débutants que la plupart des éditeurs de texte, mais dans quelque domaine que ce soit, les outils les plus puissants ont toujours réclamé plus d'apprentissage que les autres. Qui feindra de s'en étonner ?

Résumer Vim est finalement très simple :

Éditeur quelconque : «Est-ce qu'on peut ... ?» Vim : «Comment fait-on pour... ?»

Chapitre 3

Logiciel annexes

3.1 PDFLaTeX

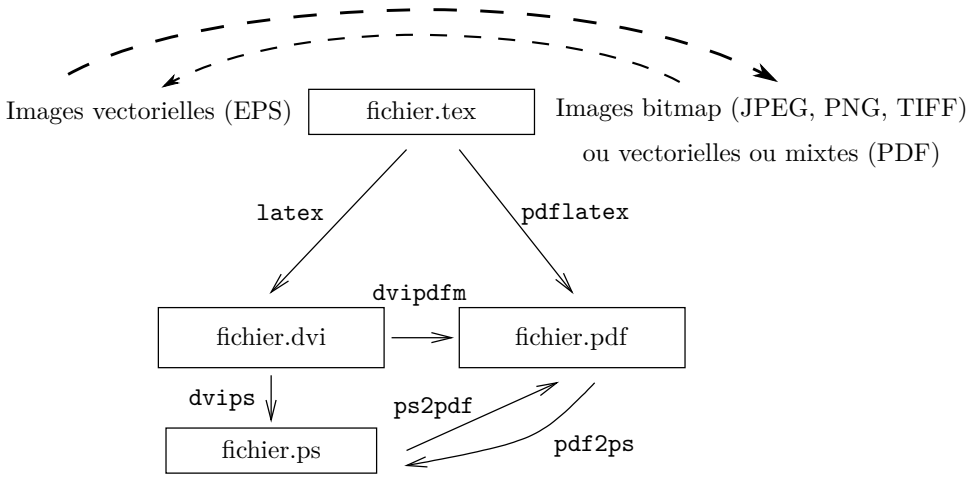
L'importance et l'universalité du PDF ne sont pas à démontrer. Bien que propriétaire, ce format est entièrement connu et documenté. Son principal avantage est qu'il permet la visualisation et l'impression avec la certitude que la mise en page sera totalement respectée.

PDFLaTeX. On ne peut pas vraiment parler de logiciel annexe : PDFLaTeX est une extension de LaTeX maintenant présente dans la plupart des distributions LaTeX, à moins de l'avoir enlevée explicitement des éléments à installer. Elle permet de produire du PDF par compilation directe du fichier source avec le plus haut niveau de qualité, sans conversion intermédiaire. Dans les documents PDF ainsi fabriqués, table des matières, liens Internet, références croisées et renvois sont cliquables.

PDFLaTeX n'est pas excessivement documentée sur Internet et on peut le regretter car, grâce à elle, un simple lecteur de PDF permet d'apprécier la qualité de LaTeX.

Le schéma suivant rappelle les deux chaînes de compilations possibles : la «traditionnelle» se faisant avec LaTeX.

Le long des flêches figurent les outils en ligne de commande permettant les compilations ou les conversions entre formats d'images. Ces outils seront utiles si on doit changer *a posteriori* de chaîne de compilation.



Comme on le voit, dans sa branche historique, LaTeX lui-même produit du DVI, qui n'est pas à proprement parler un format final, en ce sens qu'il ne permet que la visualisation. Même

une simple impression réclamera au moins une conversion au format Postscript ou... PDF, on y revient par une voie détournée.

PDFLaTeX permet l'utilisation directe d'images *bitmap* : JPEG, BMP, PNG, TIFF, etc... Il accepte aussi les illustrations vectorielles, à condition qu'elles soient au format... PDF ! Étonnant, non ?

Et contrainte commune à tous les formats : indiquer une taille ou une échelle afin d'éviter certains débordements hors de la page.

Finalement, comme l'EPS, un fichier PDF peut contenir à la fois des données vectorielles et *bitmap*, mais en présentant un poids moindre dès lors qu'il y a du *bitmap*. Un fichier JPEG, par exemple, sera inclus directement dans un fichier PDF.

À l'opposé, si on doit inclure une photo et compiler avec LaTeX, elle devra être convertie au préalable en EPS et donc donner un fichier final beaucoup plus lourd.

Autre avantage du PDF : la gestion transparente des liens internes au document, Internet, Mail, bref, des protocoles les plus courants...

Pour toutes ces raisons, j'en suis progressivement venu à ne plus travailler qu'avec PDFLaTeX que je trouve plus pratique et plus souple que LaTeX [1] et s'il n'a pas enterré son devancier pour autant, il ne faut pas s'étonner de l'intérêt croissant qu'il suscite.

Mon modèle de fichier LaTeX, grâce à l'extension `ifpdf` figurant dans son préambule, permet de compiler indifféremment avec LaTeX ou PDFLaTeX. Tout ceci, naturellement, sans aucune modification du fichier source, en laissant la possibilité, à tout instant, de changer de chaîne de compilation.

Il suffit d'avoir dans le bon dossier les images éventuelles dans les deux formats, vectoriel et *bitmap*, et de les appeler par le même nom, sans extension, au sein du fichier source :

```
\includegraphics{image0037}
```

On note bien l'absence d'extension ; si LaTeX compile, il incluera `image0037.eps` ; si pdfLaTeX compile, il incluera `image0037.pdf` ou `image0037.png` ou `image0037.jpg`.

De même qu'on n'indique pas d'extension, on peut regrouper les images dans un même sous-dossier et indiquer une fois pour toutes le chemin vers ce sous-dossier dans le préambule :

```
\graphicspath{{chemin/vers/les/images/}}
```

au lieu de l'indiquer à chaque commande `\includegraphics`. Là encore, gain de temps et meilleure lisibilité du fichier source.

3.2 Xdvi

Une fois le fichier source écrit, même partiellement, puis compilé, la première chose à faire est de visualiser le résultat. Le format de document produit à la compilation par LaTeX est le DVI [2]. Il est généralement plus léger que les formats Postscript et PDF. Ces derniers peuvent

1. Par contre, dans certains cas, notamment si on travaille avec PSTricks, une compilation traditionnelle par LaTeX reste le passage obligé.
2. *Device Independent*, indépendant du matériel comme du logiciel.

d'ailleurs être obtenus par conversion directe du DVI sans aucune perte de mise en forme ou de qualité d'impression mais ne nous égarons pas !

Avant de générer les formats finaux, on peut suivre très simplement l'évolution du document : les visionneurs DVI courants réactualisent l'affichage du document dès qu'on réactive leur fenêtre à la souris ou au clavier.

C'est notamment le cas de Xdvi, visionneur DVI que l'on trouve pratiquement dans tous les Unix. Un de mes raccourcis dans `ivsb2.vimrc` permet de le lancer.

Xdvi ou pas, tous les IDE courants permettent évidemment de lancer au moins un lecteur de DVI.

3.2.1 Fichier de configuation

Xdvi sera nécessairement lancé avec certaines options par défaut mais toutes ne vous conviendront peut-être pas. Voici les miennes :

```
XDvi*mfMode: ljfour
XDvi*pixelsPerInch: 600
XDvi*shrinkFactor: 5
XDvi*paper: a4
XDvi*wwwBrowser: firefox
XDvi*thorough: true
XDvi*shrinkButton1: 4
XDvi*shrinkButton2: 6
XDvi*shrinkButton3: 8
XDvi*expert: true
XDvi*panel*background: gray80
XDvi*noInitFile: true
XDvi*Hush: true
XDvi*hushStdout: true
```

Les options que j'ai modifiées ou ajoutées :

— «`XDvi*shrinkFactor: 5`» : place un document A4 à la française dans toute la largeur d'un écran 1024 × 768 si la fenêtre de Xdvi est maximisée ; utilisez 4 pour 1280 × 1024 ; augmentez la valeur pour réduire le grossissement ; 6 vous laissera un peu plus de place,
— «`XDvi*expert: true`» : masque la barre de boutons de Xdvi,
— «`XDvi*panel*background: gray80`» : couleur de fond de la barre de boutons de Xdvi. Pour des modifications plus fines, bonnes connaissances de X11 et de Editres requises !
— «`XDvi*noInitFile: true`» : ne pas créer de fichier de configuration de Xdvi dans les dossiers personnels,
— «`XDvi*Hush: true`» et «`XDvi*hushStdout: true`» : suppression de tous les messages d'erreurs possibles. Ces deux réglages et le précédent évitent de (trop) polluer la fenêtre de Vim.

«`XDvi*paper: a4`» et «`XDvi*wwwBrowser: firefox`» n'appellent pas d'explication. Pensez-y tout de même si votre document LaTeX, bien que francisé, n'offre pas tout à fait la présentation attendue ou si vous n'utilisez pas un format de page habituel ou le navigateur Internet indiqué.

Le fichier de configuration de Xdvi où enregistrer ces réglages peut varier suivant le système ou la distribution LaTeX :

— `/etc/X11/apps-defaults/XDvi`
— `/usr/share/texmf/xdvi/XDvi`
— `/usr/share/texmf-tetex/xdvi/XDvi`

Les habitués auront reconnu les emplacements respectifs pour teTEX 2.0.2 puis 3.0 dans Debian 3.0 Woody, puis 3.1 Sarge puis 4.0 Etch. Par :

```
$ find / -name XDvi
```

(respectez les majuscules), vous êtes à peu près sûr de trouver le vôtre.

Si vous n'avez pas les droits suffisants pour le modifier ou si vous souhaitez garder vos modifications personnelles, vous pouvez les placer dans `~/.Xdefaults` , fichier caché de votre dossier personnel et les activer par :

```
$ xrbd -merge .Xdefaults
```

3.2.2 Raccourcis-clavier

Travailler avec des raccourcis-clavier (une lettre unique, en général) est possible avec Xdvi, notamment pour atteindre une page donnée ou régler le grossissement en une fraction de seconde. Pour en savoir plus, quelle que soit votre situation : `man xdvi`. Voici ceux que j'utilise fréquemment.

n	Page suivante
p	Page précédente
37g	Aller à la page 37
<	Aller à la première page
>	Aller à la dernière page
g	Aller à la dernière page
s	Zoom page complète
5s	Zoom niveau 5
x	(Dés)active le mode *Expert*
Ctrl f	Recherche

Dernière possibilité, mais qui ne m'intéresse guère : la rétro-édition. Un double-clic dans le document DVI est censé vous ramener à l'endroit voulu dans votre éditeur de textes favori. Tous les éditeurs ne le permettent pas et des réglages supplémentaires de Xdvi et de l'éditeur sont à prévoir.

3.3 Xpdf

Xpdf est un visionneur de fichiers PDF. Son interface est plutôt spartiate mais il se montre léger et pratique. Notamment, la recompilation est possible sans fermer le fichier PDF.

Mais contrairement à Xdvi, il ne réaffiche pas automatiquement le document à l'activation de sa fenêtre ; il faut demander le réaffichage, en standard, par la touche *r*. À ce propos, les raccourcis-clavier :

r	Réaffichage
n	Page suivante
p	Page précédente
g	Aller à la page ...
Ctrl Début	Aller à la première page
Ctrl Fin	Aller à la dernière page
0 (zéro)	Zoom standard
+	Grossir
−	Réduire
w	Zoom largeur de page
z	Zoom page complète
Ctrl f	Recherche
c	Fermer la fenêtre des liens

Le fichier de configuration personnel est `~/.xpdfrc`. On peut notamment y écrire :

```
include /etc/xpdf/xpdfrc
bind t            any         toggleOutline
bind c            any         toggleContinuousMode
bind b            any         prevPage
bind p          any         print
bind p            scrLockOff  print
bind P            scrLockOff  print
bind p            scrLockOn   print
bind P            scrLockOn   print
```

En l'occurence, j'ai rajouté les seuls raccourcis qui me manquaient un peu et attribué la seule lettre `p` à la commande d'impression. J'ai trouvé `toggleOutline`[3] et la syntaxe de déclaration des raccourcis dans la page de manuel de Xpdf. RTFM, encore et toujours, donc !

```
$ man xpdf
```

Enfin, une partie des réglages figure dans le fichier `~/.Xdefaults`, également caché dans le dossier personnel :

```
xpdf.initialZoom: 150
!xpdf.paperColor: gray95
xpdf.continuousView : true
```

La validation des réglages *supplémentaires* (sous-entendu, sans annuler ceux qui existaient auparavant) se fait par la commande :

```
$ xrdb -merge .Xdefaults
```

si on ne veut pas patienter jusqu'à la prochaine déconnection/reconnection.

3. Pour activer ou désactiver la barre latérale avec les tables des matières ou signets PDF.

3.4 Evince : un autre visionneur PDF

Eh oui ! On parle d'un deuxième lecteur de fichiers PDF. Le choix d'un lecteur de PDF peut sembler sans intérêt tant l'utilisation de ce type de fichiers est devenue banale mais pour qui travaille pendant des heures avec des documents LaTeX, la question de ce choix mérite d'être posée.

Et le lecteur probablement le plus connu, Adobe Reader, mérite d'être éliminé purement et simplement des choix envisageables. En effet, il est affligé d'un défaut rédhibitoire : on ne peut pas recompiler le document qu'il est en train d'afficher !

Ce défaut fondamental éliminé, donc, n'importe lequel peut convenir, à chacun de choisir selon ses goûts en matière d'interface.

Xpdf est déjà un visionneur ultra-efficace et pour du simple visionnage, Evince n'est pas plus efficace que lui. Par contre, dès qu'il s'agit de faire autre chose, comme imprimer une sélection de pages plutôt que le document entier, imprimer dans un fichier, etc... Il se montre plus abordable, disons plutôt plus conforme aux standards actuels d'interface graphique. Son interface est en GTK et s'intègrera au mieux dans un environnement Gnome, ce qui n'empêche nullement de l'utiliser dans d'autres environnements. Un bon point, en passant : on peut l'installer sans avoir le bureau Gnome complet en dépendances.

Conforme aux standards actuels d'interface graphique, oui, mais on peut aussi travailler très vite avec lui, grâce aux raccourcis éditables à la volée : en survolant telle entrée de menu à la souris, vous pouvez fixer le raccourci de votre choix : de préférence des combinaisons de touches avec *Shift* ou *Control*[4].

Pour bénéficier de ces raccourcis éditables même hors d'un bureau Gnome, il faut avoir un fichier `.gtkrc-2.0` (caché et à la racine du dossier personnel) contenant au moins la ligne :

```
gtk-can-change-accels=1
```

À noter que depuis certaines versions, par exemple 2.30.3 dans Debian 6.0 Squeeze (stable), Evince réaffiche automatiquement un fichier PDF à chaque modification. Et là, le fan de Xpdf que je suis a fini par craquer...

Evince 3.14 : bon, là, les mecs de Gnome, ils commencent à faire leurs chiants : plus de barre de menus classique. Des actions acessibles à la souris, certes, mais plus de possibilité de changer les raccourcis-clavier à la souris par exemple.

3.5 Zathura

On peut le commander avec des raccourcis-clavier proches de ceux de Vim, en général, une seule lettre. Même en étant un inconditionnel de Vim, je dois dire que je n'en ai pas trop ressenti le besoin...

Fichier de conf sur mon site.

Mais bon... Pour de très longues séances de travail où l'on ne fait que visualiser, où l'on a besoin que de voir sa page et de la voir réaffichée automatiquement à chaque compilation, il est juste parfait. Et il n'occupe pas d'espace inutile à l'écran, c'est le moins que l'on puisse dire !

4. Les lettres uniques sont possibles comme dans Xpdf mais interfèreront immanquablement avec la fonction de recherche, dès qu'une recherche de texte contiendra ladite lettre ! Autant laisser le raccourci originel, *Ctrl f*.

Et pour imprimer, hé bien...

```
:print
```

Ah, quand même... Une interface graphique (GTK) pour imprimer...

F5 pour le mode présentation et F11 pour le mode plein écran. Bref ! Même chose que dans Evince que je fuis depuis qu'il a perdu son interface traditionnelle. Gnome 3 oblige.

Aller en début de fichier	`gg`
Aller en début de fichier	Home
Aller à la page 32	`32G`
Aller en fin de fichier	End
Aller en fin de fichier	`G`
Mode présentation	F5
Plein écran	F11
Imprimer	`:print`
Informations document	`:info`

Bref, comme il n'est pas très long, voici mon `zathurarc`. À enregistrer dans le dossier `~/.config/zathura/` .

```
set window-width 1024
set window-height 700
set adjust-open width

set guioptions ""

map <PageDown> scroll half-down
map <PageUp> scroll half-up
map b scroll half-up

map n navigate next
map p navigate previous
map h adjust_window best-fit
map w adjust_window width
map [fullscreen] h adjust_window best-fit
map [fullscreen] w adjust_window width

map <C-PageDown> navigate next
map <C-PageUp> navigate previous

# map <F11> toggle_fullscreen
```

La ligne `guioptions` vide permet d'avoir une fenêtre dépouillée de tout élément graphique.

Et comme on le voit à la fin, les commentaires sont signalés par `#` .

3.6 Manipulation de fichiers PDF

Rendez-vous à http://ivsb2.free.fr/scripts/ et à http://ivsb2.free.fr/docs/ où vous trouverez une documentation succincte sur le format PDF et deux scripts `double-demi.sh` et `sensdessudessous.sh` permettant de transformer des documents PDF pour des impressions plus économes en papier.

3.7 Xfig

Xfig est un logiciel de dessin vectoriel, c'est-à-dire que le dessin n'est pas défini point par point mais par sa forme, ce qui fait que l'on peut changer à volonté d'échelle sans faire varier la qualité du rendu du document final, ni à l'écran ni à l'impression. Le vectoriel s'oppose au *bitmap* où l'image est définie point par point et subit donc une dégradation de qualité, un effet de pixellisation, si l'on agrandit trop.

Créé vers 1988, Xfig possède, malgré son grand âge, des caractéristiques enviables par beaucoup d'autres applications. C'est l'un de ces logiciels dont disposent l'immense majorité des Unix, dont GNU/Linux et Mac OS X, et qui manquent cruellement au monde Windows.

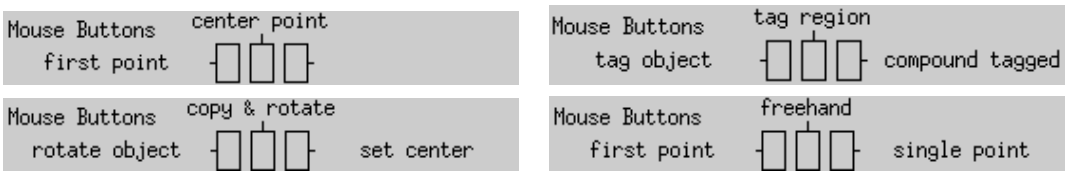
Xfig est d'une rare efficacité et donne des résultats de qualité ; il est quasi-imbattable pour les dessins au trait. Son interface très particulière, aux icônes d'une autre époque, ainsi que le rendu crénelé des dessins dans sa fenêtre ne doivent pas nous tromper : le rendu imprimé sera sans défaut.

Son mode de construction est très proche des principes des constructions de géométrie : règle, compas, rapporteur. Il est en effet possible de respecter des longueurs et des angles, pour créer des objets simples ou composés et de manipuler (déplacement, copie, rotation, etc...) lesdits objets.

Il comporte aussi deux grilles : une purement visuelle, selon les unités utilisées (système métrique ou anglo-saxon) et une «magnétique» qui évitera certains écarts peu ou pas visibles à l'écran lors de la construction. Les deux grilles ont un pas réglable et on peut les désactiver.

À l'usage, Xfig s'avère plus efficace que bien des logiciels de géométrie dynamique (pour du dessin seul, évidemment).

Il n'y a pas lieu de s'apesantir sur le maniement de Xfig : il s'apprend très vite et, au risque de me répéter, il est difficile de faire plus efficace. Il y a en fait une seule chose à savoir : le cadre «*Mouse Buttons*» en haut, à droite de l'interface de Xfig montre à tout moment le rôle des trois boutons de la souris. À surveiller de près car cela change presque à chaque fonction ! Quelques exemples :



En fait, pour gagner encore un peu en efficacité, il ne reste guère qu'à peaufiner les réglages standards de Xfig :

```
! some sample settings you may want to change
```

```
Fig.inches:     false
Fig.latexfonts:     true
Fig.zoom:       2
Fig.startgridmode:  2
```

par exemple, me permettent de mesurer en système métrique, d'utiliser directement des polices de caractères proches de celles de LaTeX et de lancer Xfig avec un grossissement et une grille 5×5 mm bien adaptés à mes petits dessins habituels.

Pour de plus amples renseignements sur les réglages possibles : `man xfig`.

Ces réglages sont à enregistrer dans le fichier `/etc/X11/apps-defaults/Fig`, si l'on souhaite diffuser les réglages à tous les utilisateurs d'une système. Si vous ne trouvez pas ce fichier à cet emplacement sur votre système, utilisez la commande :

```
$ find / -name Fig
```

Si l'on souhaite réserver ces réglages à un utilisateur individuel ou si l'on n'a pas les droits suffisants, il faudra enregistrer dans `~/.Xdefaults`. C'est également dans ces fichiers que vous devrez rajouter :

```
*customization: -color
```

si Xfig démarre avec une interface en noir et blanc mais ceci est rare avec les systèmes actuels, Debian et dérivés en particulier. Pour que les modifications soient prise en compte, tapez la commande :

```
$ xrdb -merge ~/.Xdefaults
```

Évidemment, comme cette modification est faite dans `~/.Xdefaults`, elle ne concerne qu'un utilisateur, mais pour ce type de réglages, autant laisser chaque utilisateur choisir.

Enfin, pour profiter des lettres accentuées en français, et aussi du caractère degré (°), il faut savoir que Xfig ne fonctionne pas idéalement en UTF-8. Par contre, le standard Latin 1 convient parfaitement. Pour que Xfig l'utilise, il doit être lancé en adaptant la variable d'environnement `LANG`. Au lieu de la commande `xfig &`, il faut utiliser :

```
LANG=fr_FR.iso-8859-1 xfig &
```

Cette commande peut être utilisée dans un menu de bureau, dans raccourci ou un lanceur. Elle figure maintenant dans mon `.vimrc`.

La plupart des fonctions (boutons ou entrées de menu) possèdent leur propre raccourci-clavier, que l'on peut voir en survolant les icônes à la souris. À chacun de retenir ceux qui lui sont les plus utiles.

Xfig peut exporter dans plus de 20 formats graphiques ; certains vectoriels, certains *bitmap*. Tenons-nous en à ceux utiles avec LaTeX. Dans le cas d'une compilation avec LaTeX, l'EPS est impératif ; dans le cas d'une compilation avec PDFLaTeX, on peut utiliser le PDF [5].

5. On peut exporter aussi en JPEG ou en PNG, mais quel intérêt, pour des dessins au trait, quand on dispose directement du PDF, plus léger en l'occurence ? Il se comportera alors comme du vectoriel.

Cerise à l'eau-de-vie sur le gâteau au chocolat : la double exportation EPS/PDF simultanée est disponible à partir de la version 3.2.5, du moins dans le paquet Debian de Xfig ! Ce qui permet, comme évoqué précédemment, de changer de type de compilation à tout moment sans se poser la moindre question. C'est vraiment à croire que les logiciels libres sont faits pour satisfaire les besoins *réels* de leurs utilisateurs.

Tous les autres logiciels de dessin vectoriel que j'ai pu essayer sont, au choix, propriétaires, payants, plus lourds, moins efficaces ou incapables d'exporter en EPS ou PDF. Il va sans dire qu'en général, ils cumulent la plupart de ces «qualités».

Seule lacune de Xfig, persistante : pas de copier-coller possible d'un fichier à un autre. Lacune à contourner avec les fonctions «Enregistrer sous...» ou, plutôt, «Save As...» ou bien encore «Merge», car Xfig est en anglais uniquement.

3.8 Inkscape

Bon ! Autant le dire : j'ai mis des années avant d'inclure Inkscape dans cette documentation.

Et aussi des années avant de me mettre à Inkscape. Des années dans le sens où, la flemme aidant, j'étais très efficace avec Xfig et j'avais beaucoup de mal à faire rapidement les mêmes choses avec Inkscape.

Peu-à-peu, c'est venu et je parviens enfin à avoir la même efficacité et la même rapidité.

Comme Xfig, je l'utilise principalement pour des dessins au trait et je retrouve heureusement tous les «essentiels».

Premier temps.

Ensuite, j'ai tout-de-même trouvé des avantages à Inkscape :

— le copier-coller entre documents différents, donc. Bof... C'est vrai que ça n'existait pas dans Xfig. C'est vrai que ça m'a parfois servi. Mouais...
— une gestion plus facile du texte, des polices de caractères.
— grille magnétiques et guide positionnables et réglables à volonté, ce qui n'était pas le cas dans Xfig, même si le peu de réglages disponibles me convenaient parfaitement.
— un format SVG, beaucoup plus répandu que le format Fig. Utile pour travailler un même fichier avant ou après Inkscape.

En fait, ce dernier point m'a fortement incité à passer à Inkscape. Geogebra, en particulier, avec lequel je fais des figures géométriques, exporte en SVG.

Certes, il y a un peu de «nettoyage» à prévoir quand le dessin a été récupéré dans Inkscape mais Geogebra m'est tout-de-même très utile quand des figures nécessitent des transformations géométriques évoluées : on pourrait toujours se débrouiller directement avec Inkscape ou même Xfig, c'est vrai, mais Geogebra est conçu pour faire cela, et pour le faire facilement...

J'arrête là : Inkscape a suscité des livres entiers et ces petites astuces que j'évoque sont justement l'objet d'une autre de mes documentations.

Téléchargez aide-inkscape-geogebra.pdf

Comme dans beaucoup de mes documentations, j'y ai réuni tout ce dont je tenais à me rappeler ou ce que je n'ai pas trouvé ailleurs.

3.9 Gnuplot

Non-moins historique que Xdvi, Xpdf et Xfig, Gnuplot est un logiciel de tracé de courbes (courbes de fonctions, nuages de points) en ligne de commande. On le lance par :

```
$ gnuplot
```

Son interface est une sorte de *shell* où l'on peut lancer une commande de tracé et, peu à peu, modifier les caractéristiques du graphique, toujours à l'aide de commandes. Ce mode de fonctionnement est qualifié d'interactif. Hum... Inutile de préciser que toutes les commandes doivent être validées par Entrée.

On peut aussi réunir une suite de commandes Gnuplot dans un fichier texte et lancer en console la commande :

```
$ gnuplot nom_de_fichier.gnu
```

`.gnu` est une extension facultative. Peu importe, en effet, puisqu'il s'agit d'un fichier en texte brut.

3.9.1 Nuage de points

Les nuages de points, en 2D ou 3D, peuvent être tracés à partir de fichiers textes de données, le séparateur de ces fichiers étant l'espace, les trois dimensions étant considérées dans l'ordre, sur chaque ligne.

Après avoir tapé `gnuplot` dans un terminal, on se retrouve devant un prompt signalé par le caractère `>`. On effectue le tracé par :

```
gnuplot> plot 'fichier-de-donnees.dat'
```

3.9.2 Fonctions

Pour la courbe de la fonction carrée : $f : x \longmapsto x^2$, on tape :

```
gnuplot> plot x**2
```

3.9.3 It's not a bug, it's a fitcheure !

Si vous tapez :

```
gnuplot> plot 4/3*x**2
```

vous risquez fort d'avoir une surprise, à savoir trouver sous vos zœils zébahis et zécœurés la courbe de $x \longmapsto x^2$. Normal : si vous écrivez la division `4/3` en pensant à $\frac{4}{3}$, pas de bol, Gnuplot vous retourne la partie entière de votre quotient, soit, ici, 1 !

Il faut donc prendre l'habitude d'écrire `4/3.0` ou plus court, `4/3.` (oui, 3 avec un point).

```
gnuplot> plot 4/3.*x**2
```

3.9.4 Plusieurs courbes sur le même graphique

Pas compliqué : il suffit de taper leurs équations séparées par des virgules :

```
plot x**2 , x**2/3 , 4/3*x**2 , 4/3.*x**2
```

Au passage, on notera que le problème d'écriture des divisions évoqué à l'instant ne se produit pas systématiquement : `x**2/3` est interprété correctement ; `4/3*x**2` ne l'est pas ! La notation sure est `4/3.*x**2` avec le point.

3.9.5 Tracé

Pour se restreindre à l'intervalle $[-4; 4]$, taper `set xrange [-4:4]` puis *Entrée*. On retrace la courbe par `replot` puis *Entrée*. En fait, toutes les modifications suivent le même modèle : `set machin truc` suivi de `replot` pour appliquer la modification.

Pour obtenir un repère orthonormé : `set size ratio -1`.

Pour supprimer la très peu mathématique légende en haut à droite : `unset key`.

Pour rajouter une grille : `set grid`.

Pour avoir une courbe plus lisse : `set samples 400`. Augmentez encore la valeur pour plus de précision.

Graduation des axes : `set xtics 1` pour avoir un pas de 1.

Graduations secondaires : `set mxtics 2` ou `set mxtics 4` pour diviser par 2 ou par 4 le pas précédent ; faire de même avec `mytics` pour le second axe.

3.9.6 Pour ne pas se limiter aux fonctions

Du moins en apparence...

Pour tracer la droite «verticale» d'équation $x = 3$:

```
gnuplot> plot 10000*(x-3)
```

Nécessite pour être visible de calculer de nombreux points (`set samples 10000` sur mon affichage pour ne plus créneler). Pour éviter de trop nombreux échecs ou tâtonnements, assurez-vous de limiter abscisse et ordonnée :

```
gnuplot> set xrange [-5:5] ; set yrange [-10:10]
```

À ce propos, on peut donner une suite de commandes sur la même ligne, le point-virgule jouant le rôle de séparateur.

On peut aussi définir des fonctions par leur expression et utiliser leur nom par la suite, ce qui est tout de même plus rapide si de nombreuses frappes doivent être répétées :

```
gnuplot> f(x) = exp(-x)*sin(x)
```

permet ensuite de taper :

```
gnuplot> plot [-5:5] [-1:1] f(x)
```

Ouf ! Gnuplot nous fait grâce de la notation polonaise inversée.

3.9.7 Paramétriques

```
gnuplot> set parametric
```

3.9.8 Polaires

```
gnuplot> set polar
```

3.9.9 Commandes externes

On peut aussi exécuter des commandes externes sans quitter Gnuplot :

```
gnuplot> !commande unix avec options
```

Il y a aussi quelques raccourcis-clavier pour faciliter l'édition de la ligne de commande :

Aller en début de ligne	*Ctrl a*
Aller en fin de ligne	*Ctrl e*
Effacer la ligne	*Ctrl u*
Effacer jusqu'à la fin de la ligne	*Ctrl k*
Effacer le mot précédent	*Ctrl w*

On peut aussi utiliser les flèches de déplacement.

Il y a aussi plein de choses à essayer sur la fenêtre de sortie avec la souris, que les graphes soient en deux ou trois dimensions. Comme souvent sous Unix, il ne faut pas oublier qu'une souris a trois boutons !

3.9.10 Sauvegarde

Pour sauver un travail :

```
gnuplot> save 'nom-de-fichier.gnu'
```

Pour le retrouver à la réouverture de Gnuplot :

```
gnuplot> load 'nom-de-fichier.gnu'
```

3.9.11 Exportation

Deux choses sont clairement distinguées par Gnuplot : la façon dont il produit ses données et où il les envoie. Cela peut être vérifié à tout moment par `show term` et `show out`. Dans le mode de fonctionnement normal, il vous sera répondu respectivement `x11` et `STDOUT`, c'est-à-dire le système de fenêtrage X-window et la sortie standard (la courbe est dessinée dans une fenêtre et si l'on tape la commande de sauvegarde sans plus de précision, on voit du texte défiler dans la fenêtre principale de Gnuplot).

Grâce à ce mode de fonctionnement, GNUplot peut fonctionner et produire des fichiers même sur des systèmes ne possédant pas d'interface graphique, par exemple un serveur.

Si vous désirez :

— exporter en EPS,
— vers un fichier nommé `courbe.eps`

vous devrez taper :

```
gnuplot> set term postscript eps 22 ; set out 'courbe.eps' ; replot
```

Le fait de choisir 22 indique une taille de police assez grande, utile si les EPS doivent être «agrandis» dans le document où ils seront insérés. Si vous désirez exporter au format de Xfig, afin d'enrichir ou de modifier manuellement votre courbe avec ce fameux logiciel de dessin, vous taperez plutôt :

```
gnuplot> set term fig ; set out 'courbe.fig' ; replot
```

Pour revenir au fonctionnement standard (sortie de courbes à l'écran) :

```
gnuplot> set term x11 ; set out ; replot
```

En cas de doute, on vérifie les réglages par :

```
gnuplot> show term ; show out
```

3.9.12 Aide

Je n'ai donné ici que les manipulations pratiques de base et un très faible aperçu des possibilités de Gnuplot. Vous en découvrirez plein d'autres en faisant appel à son aide intégrée :

```
gnuplot> help mot-clé
```

(le mot-clé est facultatif). Pour savoir, par exemple, quels formats d'exportation sont disponibles :

```
gnuplot> help term
```

pour savoir quels types de sortie :

```
gnuplot> help out
```

Vous pouvez quitter l'aide par *Ctrl c* et Gnuplot directement par `q`.

Chapitre 4

Installer une extension LaTeX

Nous prendrons l'exemple de l'extension Arcs, avant tout parce qu'elle est simple à installer et qu'elle consiste en un unique fichier[1]. Arcs permet la notation mathématique de l'arc de cercle : le petit arc au-dessus de deux lettres.

Vous pouvez la rechercher sur CTAN ou le télécharger directement sur mon site :

http://ivsb2.free.fr/latex/arcs.sty

Une fois le fichier dans votre dossier personnel, vous avez le choix entre deux types d'installation.

4.1 Installation système

Si vous disposez des droits de Root, vous pouvez commencer par créer l'arborescence nécessaire :

```
# mkdir -p /usr/share/texmf/tex/latex/arcs
```

puis y déposer le fichier `arcs.sty` :

```
# cp -v /home/chemin/vers/arcs.sty /usr/share/texmf/tex/latex/arcs
```

On met ensuite à jour la base de données LaTeX par :

```
# texhash
```

ou par :

```
# mktexlsr
```

1. Attention toutefois : elle est disponible dans les distributions Debian et dérivées, dans le paquet nommé `texlive-latex-extra` qui offre un grand nombre d'autres extensions. Vérifiez donc qu'elle n'est pas déjà installée sur votre système, faute de quoi la partie actuelle n'aura que peu d'intérêt. Si c'est le cas, testez donc une autre extension.

4.2 Installation personnelle

Si l'on ne dispose pas des droits de Root, on peut faire une installation dans son dossier personnel ; la démarche est strictement la même. Création de l'arborescence nécessaire dans votre dossier personnel :

```
$ mkdir -p ~/texmf/tex/latex/arcs
```

puis dépot du fichier `arcs.sty` dans ce dossier puis inscription dans la base de données LaTeX par :

```
$ texhash
```

4.3 Installation personnelle dans un dossier caché

Ça peut paraitre un peu stupide, mais voir le dossier `texmf` au milieu des documents, à la merci d'un effacement par clic de souris malencontreux me gênait quelque peu. J'ai donc renommé ce dossier en `.texmf`.

Par contre, en procédant seulement ainsi, les extensions LaTeX restaient invisibles du compilateur, même après la commande `texhash`. De plus, il fallait aussi que tous les dossiers cachés `.texmf` des dossiers personnels soient pris en compte par la distribution LaTeX. Ceci se fait par la ligne :

```
TEXMFHOME = $HOME/.texmf
```

dans le fichier `/etc/texmf/texmf.d/05TeXMF.cnf`. Le réglage sera pris en compte définitivement en lançant, en tant que Root, la commande :

```
# update-texmf
```

ou encore :

```
# update-texmf-config
```

Correctif pour Debian 7 *Wheezy* : apparemment, la modification dans `/etc/texmf/texmf.d/05TeXMF.cnf` est inopérante. Solution : dans le même dossier `/etc/texmf/texmf.d/`, placer le fichier `06local.cnf` contenant la seule ligne :

```
TEXMFHOME = ~/.texmf
```

Et pour Debian 8 Jessie : rien de tout cela. Arcs fait partie du paquet Debian `texlive-latex-extra` (de mémoire, à vérifier) et j'ai laissé tomber Picins. Avant d'y revenir. Depuis que j'ai essayé Wrapfig...

4.4 Dernière remarque importante

Je n'ai pas essayé, dernièrement de voir si la contrainte exposée dans le correctif précédent était encore d'actualité avec Debian 8 Jessie.

Quoiqu'il en soit, ne sont exposées ici que des méthodes. À chacun d'en faire une adaptation raisonnée.

4.5 Utilisation «portable»

Portable, suivant le vocable à la mode, c'est-à-dire sans installation.

Cette possibilité est curieusement assez peu connue : si un de vos fichiers LaTeX appelle l'extension Tartempion et qu'elle n'est pas installée sur votre système, vous pouvez déposer le fichier `tartempion.sty` dans le même dossier que le fichier à compiler.

Ce peut être la solution si vous laissez des fichiers à compiler à quelqu'un qui n'a pas forcément l'extension Tartempion installée sur son système.

Attention toutefois : ceci est facile si une extension tient en un unique fichier ; moins si elle est composée de nombreux fichiers ou dossiers.

Chapitre 5

Fichiers à télécharger

Je les mets parfois à jour. Pour obtenir les dernières versions, vous pouvez visiter mon site :

http://ivsb2.free.fr/

et en particulier :

http://ivsb2.free.fr/vim
http://ivsb2.free.fr/latex

5.1 Modèles LaTeX

modele.tex : mon modèle de base. Documents généraux et pouvant comporter des mathématiques assez évoluées. Des marges à 2 cm, en particulier, donc nettement réduites par rapport aux standards LaTeX, initialement prévus pour le masssicotage.

ex.tex : variante, avec des marges plus petites encore. Plutôt prévu pour des documents compactés sur une page, sauf à accepter des hauts de page très réduits à partir de la page 2.

presentation.tex : modèle de présentation au format PDF avec LaTeX-Beamer.

Dans chacun de ces fichiers, beaucoup d'options pratiques brièvement expliquées et immédiatement utilisables : il suffit de les décommenter pour les utiliser.

Si vous utilisez un Unix quelconque, je vous suggère de les enregistrer tels que, en conservant leurs noms, dans un sous-dossier caché du dossier personnel (`~/.latex` chez moi). La configuration de Vim que je fournis pointe vers ce chemin relatif.

5.2 Pour Vim

ivsb2.vimrc : mes réglages pour Vim et gVim. Attention : comme expliqué précédemment, il ne constitue pas un `vimrc` complet mais un complément à signaler dans le `vimrc` habituel par la commande `source`.

ouin-ouin.vimrc : même chose pour gVim pour MS Windows, testé sur les versions 98, 2000 et XP Professionnel. Vérifiez aussi la présence dans le `_vimrc` original (j'ignore pourquoi il y a un *underscore*) d'une ligne du genre :

```
behave xterm
```

Dernière précaution : assurez-vous d'adapter les chemins vers les applications et les modèles utilisés.

blabbla

5.3 Makefile

Utile pour des projets avec mise-à-jour d'un ou plusieurs dossiers de documentation, par exemple, ou si on ne souhaite pas louper la recompilation suite à la modification d'un seul fichier perdu au milieu des autres...

Bref ! Pas trop le temps de développer... Si vous savez ce qu'est un Makefile, voici le mien à toutes fins zutiles...

Makefile-latex

5.4 Ce document

edition-latex.pdf est mis à jour plusieurs fois par an. D'autres documentations sont disponibles à http://ivsb2.free.fr/docs/