# HOWTO de l'éditeur ViM couleur (Vi aMélioré, avec coloration syntaxique)

Al Dev (Alavoor Vasudevan) *alavoor@yahoo.com* ;
Version française par Arnaud Launay, `asl@launay.org`
v14.0, 16 Août 2000

Ce document est un guide pour configurer très rapidement l'éditeur couleur ViM sur les systèmes Linux ou Unix. Les informations présentées ici augmenteront la productivité du programmeur puisque l'éditeur ViM supporte la coloration syntaxique et les fontes grasses qui augmentent la « lisibilité » du code. La productivité d'un programmeur est accrue de 2 à 3 fois avec un éditeur couleur comme ViM. Les informations de ce document s'appliquent à tous les systèmes d'exploitation sous lesquels Vim fonctionne, c'est-à-dire Windows 95/NT, Apple Mac, et toutes les versions d'Unix telles que Linux, FreeBSD, Solaris, HPUX, AIX, SCO, Ultrix, Sinix, BSD, SCO, etc. (ce qui revient à dire, à peu près tous les OS de la planète !).

## Contents

# 1 Introduction

L'éditeur ViM signifie « Vi iMproved » (Vi amélioré). Vi est l'éditeur le plus populaire et le plus puissant du monde Unix. Son nom vient de l'abbréviation éditeur ***Vi***suel. Un éditeur visuel comme Vi était un grand progrès par rapport aux éditeurs en ligne comme 'ed' (ou 'ex'). Les éditeurs 'ed' et 'ex' sont toujours disponibles sous Linux : voyez 'man ed' et 'man ex'.

Un bon éditeur augmentera la productivité du programmeur. Vim supporte la coloration syntaxique du code ainsi que différentes fontes, normales, grasses ou italiques. Les éditeurs couleurs comme ViM augmentent la productivité du programmeur de 2 à 3 fois ! Les programmeurs peuvent lire le code beaucoup plus rapidement si la syntaxe du code est colorées et mise en évidence.

## 1.1 Avant d'installer

Avant d'installer ViM, référez vous aux notes relevant de l'OS et aux informations sur la compilation et l'usage de ViM sur -

- Allez ici et regardez les fichiers os_*.txt `http://cvs.vim.org/cgi-bin/cvsweb/vim/runtime/doc`

Si vous n'avez pas le paquetage ViM (RPM, DEB, tar, zip) alors chargez le code source par ftp sur le site officiel de ViM

- La page maison de ViM est sur `http://www.vim.org`
- Le site miroir US est sur `http://www.us.vim.org`
- Le site FTP est sur `ftp://ftp.vim.org/pub/vim`
- Ou utilisez un des miroirs sur `ftp://ftp.vim.org/pub/vim/MIRRORS`

## 1.2 Installer ViM sous RedHat Linux

Pour utiliser ViM, installez les paquetages rpm suivants sous RedHat -

```
        rpm -i vim*.rpm
Ou comme ceci -
        rpm -i vim-enhanced*.rpm
        rpm -i vim-X11*.rpm
        rpm -i vim-common*.rpm
        rpm -i vim-minimal*.rpm
```

Vous pouvez voir la liste des fichiers ViM que rpm a installé par

```
        rpm -qa | grep ^vim | xargs rpm -ql | less
ou
        rpm -qa | grep ^vim | awk '{print "rpm -ql " $1 }' | /bin/sh | less
```

Et regardez la sortie en utilisant j, k, CTRL+f, CTRL+D, CTRL+B, CTRL+U ou les touches fléchées, page up/down. Voyez aussi 'man less'.

Notez que les paquetages RPM pour RedHat Linux utilisent une interface Motif. Si vous avez installé les librairies GTK sur votre système, vous pouvez envisager de recompiler ViM à partir du source code afin de bénéficier d'une interface graphique propre. Pour les informations sur la compilation du code de ViM, voyez « installer ViM sur Unix » plus bas.

## 1.3 Installer ViM sous GNU Debian Linux

Pour installer ViM sous Debian Linux (GNU Linux), identifiez vous en tant que superutilisateur et lorsque vous êtes connecté à Internet tapez -

```
apt-get install vim vim-rt
```

Cela chargera la dernière version de ViM, l'installera, le configurera, et supprimera les fichiers .deb chargés. Le premier paquetage listé est ViM, l'éditeur standard, compilé avec support pour X11, vim-rt est le vim-runtime, c'est-à-dire les fichiers de syntaxe et d'aide.

## 1.4 Installer ViM sous Unix

Pour les autres versions d'Unix comme Solaris, HPUX, AIX, Sinix, SCO, récupérez les fichiers sources (voyez 1.1 ())

```
        zcat vim.tar.gz | tar -xvf -
        cd vim-5.6/src
        ./configure --enable-gui=motif
        make
        make install
```

## 1.5 Installer ViM sous Microsoft Windows 95/NT

Pour Windows 95/NT, récupérez les fichiers zip et installez-les en cliquant sur setup. Vous devez charger DEUX fichiers zip -

- Fichier contenant le runtime **vim*rt.zip**
- Fichier de commande ViM **vim*56.zip** dont la version est la 5.6

Récupérez ces deux fichiers (voir 1.1 ())

Décompactez les fichiers zip en utilisant Winzip `http://www.winzip.com` . Les deux fichiers zip (vim*rt.zip et vim*56.zip) doivent être décompressés dans le même répertoire, disons **c:\vim**.

Pour Windows 95/98, ajoutez la variable d'environnement VIM dans autoexec.bat en ajoutant cette ligne -

```
set VIM=c:\vim\vim56
```

Pour Windows NT, ajoutez la variable d'environnement dans le dialogue **Control Panel** | **System** | **Environment** | **System Properties** :

```
VIM=c:\vim\vim56
```

La variable VIM doit pointer là où vous avez installé le répertoire vim56. Vous pouvez aussi ajouter l'emplacement de gvim.exe à votre PATH.

Il est probable que vous deviez vous déconnecter et vous réidentifier pour avoir les bonnes variables. À l'invite MSDOS tapez -

```
        set vim
```

Et vous devriez voir - VIM=c:\vim\vim56

Créez un raccourci sur votre bureau en copiant/collant de c:\vim\vim56\gvim.exe. Copiez le fichier gvimrc_example vers $VIM_gvimrc. Dans mon cas, il s'agit de c:\vim\vim56_gvimrc.

## 1.6 Installer ViM sous VMS

### 1.6.1 Charger les fichiers

Vous aurez besoin des archives Unix et extra pour construire vim.exe pour VMS. Pour utiliser toute la puissance de ViM vous aurez également besoin des fichiers runtime. Prenez ces fichiers (voir 1.1 ()).

Vous pouvez charger des exécutables précompilés sur `http://www.polarfox.com/vim` .

Les auteurs de ViM VMS sont -

- *zoltan.arpadffy@essnet.se*
- *arpadffy@altavista.net*
- *cec@gryphon.gsfc.nasa.gov*
- *BNHunsaker@chq.byu.edu*
- *sandor.kopanyi@altavista.net*

### 1.6.2 Compiler

Décompactez les archives Unix et Extra dans un même répertoire. Dans le sous-répertoire <.SRC> vous devriez trouver le fichier make OS_VMS.MMS. En éditant ce fichier vous pourrez choisir les versions des caractères, des interfaces et du débogage. Il y a également des options additionnelles concernant les supports pour Perl, Python et Tcl.

Vous aurez besoin soit de l'utilitaire DECSET mms ou du clone disponible gratuitement appellé mmk (VMS n'a pas d'utilitaire make en distribution standard). Vous pouvez récupérer mmk sur http://www.openvms.digital.com/freeware/MMK/

Si vous avez MMS sur votre système, la commande

> mms /descrip=os_vms.mms

construira votre version personnalisée de ViM. La commande équivalente pour mmk est :

> mmk /descrip=os_vms.mms

### 1.6.3 Déploiement

ViM utilise une structure de répertoires spéciale pour les fichiers de documentation et d'utilisation :

```
vim (ou autre)
 |- tmp
 |- vim55
 |----- doc
 |----- syntax
 |- vim56
 |----- doc
 |----- syntax
 vimrc    (fichiers système rc)
 gvimrc
```

Utilisez :

```
>       define/nolog device:[leading-path-here.vim]       vim
>       define/nolog device:[leading-path-here.vim.vim56] vimruntime
>       define/nolog device:[leading-path-here.tmp]       tmp
```

pour que vim.exe puisse trouver ses fichiers de documents, ses types de fichiers et de syntaxe, et pour spécifier un répertoire où les fichiers temporaires seront placés. Copiez le sous-répertoire « runtime » de la distribution vim dans vimruntime.

Note : les variables $VIMRUNTIME et $TMP sont optionnelles. Lisez en plus sur :help runtime.

### 1.6.4 Usage pratique

Habituellement vous devrez faire tourner une seule version de ViM sur votre système, il est donc suffisant de dédier un seul répertoire à ViM. Copiez toute la structure des répertoires de runtime dans la position de déploiement. Ajoutez les lignes suivantes à votre LOGIN.COM (dans le répertoire SYS$LOGIN). Définissez la variable $VIM en tant que :

```
>       $ define VIM device: <path>
```

Configurez quelques symboles :

```
>       $ ! vi lance ViM en mode caractère
>       $ vi*m  :== mcr device:<path>VIM.EXE

>       $ !gvi lance ViM en mode GUI
>       $ gv*im :== spawn/nowait mcr device:<path>VIM.EXE -g
```

Créez les fichiers .vimrc et .gvimrc dans votre répertoire personnel (SYS$LOGIN).

La méthode la plus simple est de renommer les fichiers d'exemples. Vous pouvez laisser le fichier de menu (MENU.VIM) et les fichiers vimrc et gvimrc dans le répertoire original $VIM. Ce sera la configuration par défaut pour tous les utilisateurs, mais ceux-ci pourront apporter leurs propres modifications à la configuration via les fichiers .vimrc et .gvimrc de leur répertoire personnel. Ceci devrait marcher sans problème.

Note : Rappellez-vous, les fichiers systèmes rc (défaut pour tous les utilisateurs) n'ont pas de « . » final. Ainsi, les fichiers systèmes rc sont :

```
>       VIM$:vimrc
>       VIM$:gvimrc
>       VIM$:menu.vim
```

et les fichiers utilisateurs personnalisés sont :

```
>       sys$login:.vimrc
>       sys$login:.gvimrc
```

Vous pouvez vérifier que tout fonctionne et est à la bonne place avec la commande :version.

```
Exemple de LOGIN.COM :

>       $ define/nolog VIM RF10:[UTIL.VIM]
>       $ vi*m  :== mcr VIM:VIM.EXE
>       $ gv*im :== spawn/nowait mcr VIM:VIM.EXE -g
>       $ set disp/create/node=192.168.5.223/trans=tcpip
```

Note : Cette configuration devrait être suffisante si vous travaillez sur un serveur seul ou dans un environnement clusterisé, mais si vous désirez utiliser ViM en tant qu'éditeur internoeuds, il suffit de définir le « chemin » complet :

```
>       $ define VIM "<server_name>[""user password""]::device:<path>"
>       $ vi*m :== "mcr VIM:VIM.EXE"
```

par exemple :

```
>       $ define VIM "PLUTO::RF10:[UTIL.VIM]"
>       $ define VIM "PLUTO""ZAY mypass""::RF10:[UTIL.VIM]" ! si un pass est nécessaire
```

Vous pouvez aussi utiliser la variable $VIMRUNTIME pour pointer sur la bonne version de ViM si vous avez plusieurs versions installées en même temps. Si $VIMRUNTIME n'est pas défini ViM prendra la valeur de la variable $VIM. Vous pourrez trouver plus d'informations sur la variable $VIMRUNTIME en tapant :help runtime en commande ViM.

### 1.6.5 Questions sur le mode GUI

VMS n'est pas un environnement X window natif, vous ne pouvez donc pas lancer ViM en mode GUI « juste comme ça ». Mais ce n'est pas trop compliqué d'obtenir un ViM fonctionnel.

```
1) Si vous travaillez sur la console X VMS.
   Lancez ViM avec la commande :

>       $ mc device:<path>VIM.EXE -g

ou tapez :gui en commande à l'invite ViM. Pour plus d'infos tapez :help gui

2) Si vous travaillez sur un autre environnement X window comme Unix ou une
console VMS X distante. Configurez votre affichage sur cet hôte avec :

>       $ set disp/create/node=<addresse IP>/trans=<nom-transport>

et lancez ViM comme au point 1. Vous pourrez trouver plus d'aide dans la
documentation VMS ou tapez: help set disp à l'invite VMS.
   Exemples :

>       $ set disp/create/node=192.168.5.159             ! transport par défaut DECNet
>       $ set disp/create/node=192.168.5.159/trans=tcpip ! réseau TCP/IP
>       $ set disp/create/node=192.168.5.159/trans=local ! affichage sur le même noeud
```

Note : Vous ne devez en définir qu'un. Pour plus d'infos tapez $help set disp à l'invite VMS.

## 1.7 Installer ViM sous OS/2

Lisez les notes de version de ViM sur OS/2, voyez 1.1 ().

Pour le moment il n'y a pas de version PM native de la version GUI de ViM ; la version OS/2 est une application en console. Néanmoins, il y a maintenant une version Win32s-compatible GUI, qui devrait être utilisable par les utilisateurs de Warp4 (qui supporte Win32s) dans une session Win-OS/2. Les notes de ce fichier se réfèrent à la version console native.

Pour utiliser ViM, vous aurez besoin de l'environnement de lancement emx (au moins la version 0.9b). Elle est généralement disponible en tant que (demandez le à Archie) :

```
    emxrt.zip     emx runtime package
```

## 1.8 Installer ViM sur Apple Macintosh

Lisez les notes de version de ViM sur Mac, voyez 1.1 ().

L'auteur de ViM sur Mac (de l'ancienne version 3.0) est

```
Eric Fischer
5759 N. Guilford Ave
Indianapolis IN 46220 USA
```

Écrivez à *enf@pobox.com*

Rapport de Bogue Mac. Lorsque vous avez à rapporter tout changement spécifique au Mac, bogue ou option, incluez l'adresse suivante dans le champ « To: » ou « Copy To: ». *dany.stamant@sympatico.ca*

ViM compile sans ajout avec le projet CodeWarrior en utilisant CodeWarrior 9. Si vous utilisez une version plus récente (CW Pro) vous devrez convertir le projet avant toute chose. Pour compiler ViM pour Macs 68k vous devrez ouvrir la ressource « size » dans ResEdit et activer le bouton « High level events aware » pour avoir un copier/coller fonctionnel. Vous devrez augmenter la partition de mémoire à au moins 1024 koctets pour éviter à ViM de se crasher faute de mémoire suffisante.

# 2 Configurez les fichiers d'initialisation de ViM

Pour permettre la coloration syntaxique, vous DEVEZ copier le fichier vimrc dans votre répertoire personnel. Il ajoutera également le menu « Syntax » pour la commande gvim. Vous pouvez cliquer sur le menu Syntax et sélectionner le langage approprié, comme C++, Perl, Java, SQL, ESQL, etc.

```
cd $HOME
cp /usr/doc/vim-common-5.7/gvimrc_example  ~/.gvimrc
cp /usr/doc/vim-common-5.7/vimrc_example  ~/.vimrc
```

Les commentaires du .vimrc commencent avec les apostrophes ("). Vous pouvez personnaliser vim en éditant le fichier $HOME/.vimrc et en rajoutant les lignes suivantes :

```
set guifont=8x13bold
"set guifont=9x15bold
"set guifont=7x14bold
"set guifont=7x13bold
```

Il est **extrêmement** recommandé que vous mettiez les compteurs « tabstop » et « shiftwidth » à 4. Le compteur « tabstop » est le nombre d'espaces que TAB ajoutera lorsque vous éditerez sous vim. Le compteur « shiftwidth » est le nombre d'espaces qui décaleront les lignes en tapant les commandes vi ">>" ou "<<". Référez vous au tutorial de Vim 8 () pour plus de détails. Pour mettre en place tabstop et shiftwidth :

```
set tabstop=4
set shiftwidth=4
set nowrapscan
set ignorecase
```

Pour voir la liste des fontes disponibles sous Linux/Unix voyez la commande **xlsfonts**. Tapez -

```
bash$ xlsfonts | less
bash$ xlsfonts | grep -i bold | grep x
bash$ man xlsfonts
```

## 2.1 Paramètres du Xdefaults

Vous pouvez configurer quelques-unes des propriétés de Vim dans le fichier Xdefaults.

**ATTENTION :** *Ne mettez pas* ***Vim*geometry****, il coincerait les menu gvim, utilisez plutôt* ***Vim.geometry*** *à la place.*

Éditez votre $HOME/.Xdefaults et ajoutez les lignes suivantes :

```
! GVim super couleurs.
Vim*useSchemes:         all
Vim*sgiMode:            true
Vim*useEnhancedFSB:     true
Vim.foreground:         Black
!Vim.background:        lightyellow2
Vim*background:         white
! N'utilisez PAS Vim*geometry, il coincerait les menu gvim,
! utilisez Vim.geometry. Un astérisque entre Vim et geometry n'est pas autorisé.
! Vim.geometry: widthxheight
Vim.geometry:           88x40
!Vim*font:              -misc-fixed-medium-r-normal--20-200-75-75-c-100-iso8859-15-*5
Vim*menuBackground: yellow
Vim*menuForeground: black
```

Afin que ces changements soient pris en compte, tapez :

```
        xrdb -merge $HOME/.Xdefaults
        man xrdb
```

Vous pouvez aussi éditer votre fichier /.gvimrc pour changer les couleurs de fond :

```
        gvim $HOME/.gvimrc
Les meilleures couleurs de fond sont jaune clair ou blanc.
        highlight Normal guibg=lightyellow
```

## 2.2 Fichier vimrc d'exemple

Vous pouvez changer des paramètres comme la couleur, les fontes grasses ou normales dans le fichier gvimrc. Il est **extrêmement** recommandé de configurer la couleur de fond au jaune clair ou blanc. Les ergonomistes disent que la meilleure couleur de fond est le jaune clair ou le blanc. Vous pouvez changer la variable « guibg » comme suit :

```
        highlight Normal guibg=lightyellow
```

Le fichier d'exemple de vim-5.5/runtime/vimrc_example est celui-ci :

```
" Un exemple de fichier vimrc.
"
" Mainteneur :  Bram Moolenaar <Bram@vim.org>
" Dernières modifications : 9 Sep 1999
"
```

```
" Pour l'utiliser, copiez le dans
"     pour Unix et OS/2 :  ~/.vimrc
"            pour Amiga :  s:.vimrc
" pour MS-DOS and Win32 :  $VIM\_vimrc

set nocompatible        " Utilise les défauts Vim (bien mieux !)
set bs=2                " autorise l'effacement de tout en mode insertion
set ai                  " toujours utiliser l'autoindentation
set backup              " Conserver un fichier de sauvegarde
set viminfo='20,\"50    " Lit/écrit un fichier .viminfo, ne sauve pas plus
                        " de 50 lignes de registres
set history=50          " Conserve 50 lignes d'historique des commandes
set ruler               " Montre toujours la position du curseur

" Pour l'interface Win32: retirez l'option 't' de 'guioptions': pas d'entrée menu tearoff
" let &guioptions = substitute(&guioptions, "t", "", "g")

" N'utilise pas le mode Ex, utilise Q pour le formatage
map Q gq

" p en mode Visuel remplace le texte sélectionné par le registre "".
vnoremap p <Esc>:let current_reg = @"<CR>gvdi<C-R>=current_reg<CR><Esc>

" Active la coloration syntaxique lorsque le terminal dispose de couleurs
" Active aussi la coloration de la dernière chaîne recherchée.
if &t_Co > 2 || has("gui_running")
  syntax on
  set hlsearch
endif

" Ne lance la partie suivante que si le support des autocommandes a été inclus
" lors de la compilation
if has("autocmd")

 " Dans les fichiers textes, toujours limiter la longueur du texte à 78
 " caractères
 autocmd BufRead *.txt set tw=78

 augroup cprog
  " Supprime toutes les autocommandes cprog
  au!

  " Lors du début d'édition d'un fichier :
  "   Pour les fichiers C et C++ active le formatage des
  "   commentaires et l'indentation C
  "   Pour les autres fichiers, les désactive.
  "   Ne pas changer l'ordre, il est important que la ligne
  "   avec * arrive avant.
  autocmd FileType *      set formatoptions=tcql nocindent comments&
  autocmd FileType c,cpp  set formatoptions=croql cindent comments=sr:/*,mb:*,el:*/,://
 augroup END

 augroup gzip
  " Supprime toutes les autocommandes gzip
  au!
```

```
 " Active l'édition des fichiers gzippés
 " Active le mode binaire avant de lire le fichier
 autocmd BufReadPre,FileReadPre        *.gz,*.bz2 set bin
 autocmd BufReadPost,FileReadPost      *.gz call GZIP_read("gunzip")
 autocmd BufReadPost,FileReadPost      *.bz2 call GZIP_read("bunzip2")
 autocmd BufWritePost,FileWritePost    *.gz call GZIP_write("gzip")
 autocmd BufWritePost,FileWritePost    *.bz2 call GZIP_write("bzip2")
 autocmd FileAppendPre                 *.gz call GZIP_appre("gunzip")
 autocmd FileAppendPre                 *.bz2 call GZIP_appre("bunzip2")
 autocmd FileAppendPost                *.gz call GZIP_write("gzip")
 autocmd FileAppendPost                *.bz2 call GZIP_write("bzip2")

 " Après la lecture du fichier compressé : décompresse le texte dans le
 " buffer avec "cmd"
 fun! GZIP_read(cmd)
   let ch_save = &ch
   set ch=2
   execute "'[,']!" . a:cmd
   set nobin
   let &ch = ch_save
   execute ":doautocmd BufReadPost " . expand("%:r")
 endfun

 " Après l'écriture du fichier compressé : compresse le fichier écrit avec "cmd"
 fun! GZIP_write(cmd)
   if rename(expand("<afile>"), expand("<afile>:r")) == 0
     execute "!" . a:cmd . " <afile>:r"
   endif
 endfun

 " Avant l'ajout au fichier compressé : décompresser le fichier avec "cmd"
 fun! GZIP_appre(cmd)
   execute "!" . a:cmd . " <afile>"
   call rename(expand("<afile>:r"), expand("<afile>"))
 endfun

augroup END

" Ce qui suit est désactivé, car il change la liste de sauts. On ne peut pas utiliser
" CTRL-O pour revenir en arrière dans les fichiers précédents plus d'une fois.
if 0
 " Lors de l'édition d'un fichier, saute toujours à la dernière position du curseur.
 " Ceci doit se trouver après les commandes de décompression.
  autocmd BufReadPost * if line("'\"") && line("'\"") <= line("$") | exe "normal `\"" | endif
endif

endif " has("autocmd")
```

## 2.3 Fichier gvimrc d'exemple

L'exemple de gvimrc de vim-5.5/runtime/gvimrc_example ressemble à celui-ci :

---

```
" Un exemple de fichier gvimrc.
```

```
" Ces commandes sont exécutées lors du lancement de l'interface graphique.
"
" Mainteneur :  Bram Moolenaar <Bram@vim.org>
" Dernières modifications : 2 Fév 1999
"
" Pour l'utiliser, copiez le dans
"     pour Unix et OS/2 :  ~/.gvimrc
"            pour Amiga :  s:.gvimrc
"  pour MS-DOS and Win32 :  $VIM\_gvimrc

" Passe les commandes externes par un tuyau au lieu d'un pseudo-tty
"set noguipty

" Active la fonte X11 à utiliser
" set guifont=-misc-fixed-medium-r-normal--14-130-75-75-c-70-iso8859-1

" Rend la ligne de commande de 2 lignes plus grande
set ch=2

" Permet le shift-insert fonctionnel comme dans les Xterm
map <S-Insert> <MiddleMouse>
map! <S-Insert> <MiddleMouse>

" Ne fait ceci que pour Vim de version 5.0 et ultérieures.
if version >= 500

  " J'aime avoir des chaînes éclairées dans les commentaires C
  let c_comment_strings=1

  " Active la coloration syntaxique.
  syntax on

  " Active la coloration de la chaîne recherchée.
  set hlsearch

  " Pour la version Win32, on a "K" qui cherche le keyword dans un fichier d'aide
  "if has("win32")
  "  let winhelpfile='windows.hlp'
  "  map K :execute "!start winhlp32 -k <cword> " . winhelpfile <CR>
  "endif

  " Cache le pointeur de souris lorsque l'on tape
  set mousehide

  " Utilise des couleurs sympathiques
  " Le fond pour le texte normal est en gris clair
  " Le texte sous la dernière ligne est en gris sombre
  " Le curseur est gris
  " Les constantes ne sont pas soulignées mais ont un fond légèrement plus clair
  highlight Normal guibg=grey90
```

```
  highlight Cursor guibg=Green guifg=NONE
  highlight NonText guibg=grey80
  highlight Constant gui=NONE guibg=grey95
  highlight Special gui=NONE guibg=grey95

endif
```

# 3 Fichier d'initialisation de la coloration syntaxique

## 3.1 Méthode automatique

La section ci-dessous provient d'une session gvim en tapant « :help syntax » -

```
bash$ gvim un_test
:help syntax
```

Cliquez sur le menu Window=>Close_Others pour fermer les autres fenêtres. Utilisez ensuite CTRL+] du menu « Procédures de chargement de syntaxe » qui vous emmènera sur la bonne entrée. Utilisez CTRL+T pour revenir en arrière.

Si un type de fichier que vous désirez utiliser n'est pas encore détecté, il y a deux moyens pour l'ajouter. Il vaut mieux ne pas modifier le fichier *$VIMRUNTIME/filetype.vim*. Il sera réécrit lors de l'installation d'une nouvelle version de ViM. Crééz un fichier dans $HOME/vim/myfiletypes.vim et ajoutez lui ces lignes -

```
" Nom du fichier : $HOME/vim/mestypesdefichiers.vim
" mestypesdefichiers
augroup filetype
        au! BufRead,BufNewFile *.mine   set filetype=mine
        au! BufRead,BufNewFile *.xyz    set filetype=drawing
        au! BufRead,BufNewFile *.prc    set filetype=plsql
augroup END
```

Ajoutez ensuite une ligne à vos fichiers $HOME/.vimrc et $HOME/.gvimrc pour initialiser la variable « mestypesdefichiers » au nom de ce fichier. (**ATTENTION :** Vous DEVEZ mettre ceci dans les deux fichiers vimrc et gvimrc pour que ceci fonctionne. Exemple :

```
<code>
        let myfiletypefile = "~/vim/myfiletypes.vim"
```

**NOTE :** Assurez vous que vous initialisez bien « mestypesdefichiers » avant de passer à la détection du type de fichier. Ceci doit intervenir avant toute commande « :filetype on » ou « :syntax on ».

Votre fichier sera alors parcouru après l'intallation des autocommandes de type de fichier par défaut. Ceci vous permet de passer outre tous les paramètres par défaut, en utilisant « :au! » pour supprimer les autocommandes de type de fichier existant pour le même schéma. Seule l'autocommande pour parcourir le fichier scripts.vim est donné plus tard. Ceci vous permettra de vous assurer que les autocommandes dans « mestypesdefichiers » sont utilisées avant de vérifier le contenu du fichier.

### 3.2 Méthode manuelle

Au lieu d'utiliser un menu « Syntax » vous pouvez lire manuellement le fichier de syntaxe. Éditez le fichier avec gvim et donnez la commande « so » à : (en mode d'échappement). Par exemple :

```
        gvim foo.pc
        :so $VIM/syntax/esqlc.vim
```

Les fichiers de syntaxe sont dans /usr/share/vim/syntax/*.vim. Vim supporte plus de 120 fichiers de syntaxe différents pour divers langages comme C++, PERL, VHDL, JavaScript, etc., et énormément d'autres !

Chaque fichier de syntaxe supporte une ou plusieurs extensions de fichiers par défaut, par exemple, le fichier de syntaxe JavaScript supporte l'extension *.js. Si vous utilisez une extension qui crée un conflit avec un autre fichier de syntaxe par défaut (comme ajouter du JavaScript à un fichier *.html) vous pourrez alors charger le fichier de syntaxe additionnel avec la commande :so $VIM/syntax/javascript.vim. Pour éviter d'avoir à le taper, vous pouvez créer un lien symbolique par :

```
        ln -s $VIM/syntax/javascript.vim js
        gvim foo.html  (... this file contains javascript functions and HTML)
        :so js
```

## 4 Usage de ViM

Vous pouvez utiliser ViM sous deux modes, l'un avec interface graphique et l'autre sans. Pour utiliser l'interface graphique utilisez la commande :

```
        gvim foo.cpp
```

Pour utiliser le mode non-graphique utilisez :

```
        vim foo.cpp
ou le mode ancien
        vi foo.cpp
```

Il est très recommandé que vous utilisiez toujours gvim à la place de vim, car le mode GUI avec les couleurs augmentera réellement votre productivité.

Le mode GUI gvim permet ce qui suit :

- vous pouvez marquer le texte en utilisant la souris pour faire des copier/couper/coller ;
- vous pouvez utiliser la barre de menu qui a les boutons File, Edit, Window, Tools, Syntax et Help ;
- également dans un futur proche dans gvim - une seconde barre de menu affichera la liste des fichiers en cours d'édition, et vous pourrez changer entre les fichiers en cliquant sur leur nom, à moins que vous n'utilisiez les commandes vi - :e#, :e#1, :e#2, :e#3, :e#4, etc. pour sélectionner les fichiers.

# 5 Compagnons Vi

En général ViM est utilisé en conjonction avec d'autres outils puissants comme **ctags** et **gdb**. **ctags** est très rapide pour la navigation au milieu de millions de lignes de code « C/C++ » et **gdb** est pour le déboguage de code « C/C++ ». Une brève introduction à ces deux commandes indispensables sera donnée dans ce chapitre.

**ctags** est la commande la plus puissante pour coder en C, C++, Java, Perl, scripts shell Korn/Bourne ou Fortran. Les développeurs utilisent intensivement **ctags** pour naviguer au travers de milliers de fonctions à l'intérieur de programmes C/C++. Voyez 'man ctags' sous Unix. Il est **très important** que vous appreniez comment utiliser ctags pour développer des programmes en C, C++, Java, etc. La navigation est la tâche simple la plus importante lors de développement en C ou C++. L'utilisation de ctags peut vous aider à rapidement lire le code en sautant de la ligne d'appel à la fonction appellée, en s'enfonçant dans les appels de fonctions imbriquées, et en remontant de la fonction la plus imbriquée jusqu'à la fonction principale. Vous pouvez aller et revenir de fonction en fonction très rapidement.

Sans NAVIGATION vous serez complètement perdu ! **ctags** est comme le COMPAS magnétique nécessaire aux programmeurs.

Utilisation de **ctags** :

```
ctags *.cpp
gvim -t foo_function
gvim -t main
```

Ceci éditera le fichier programme C++ qui contient la fonction foo_function() et placera directement le curseur sur la première ligne de la fonction foo_function(). La deuxième commande vous placera sur la ligne contenant la définition de la fonction main().

À l'intérieur de l'éditeur ViM, vous pouvez sauter à une fonction en tapant : (double point) tag nom_de_la_fonction comme ci dessous :

```
:tag fonction_exemple
```

Ceci placera le curseur sur la première ligne de fonction_exemple().

Si vous voulez sauter dans la fonction à partir de la ligne du fichier contenant le nom de la fonction, placez le curseur juste avant le nom de la fonction et tapez **CTRL+]** (tapez la touche de contrôle et le crochet gauche simultanément).

```
// code d'exemple
switch(id_number) {
        Case 1:
                if ( foo_function( 22, "abcef") == 3 )
                    ^
                    |
                    |
                    |
  Placez le curseur ici (juste avant foo_function) et tapez CTRL+]
  Ceci vous emmènera à la fonction nommée "foo_function".
  Pour revenir à cette ligne tapez CTRL+t
```

Pour revenir à la ligne d'appel tapez **CTRL+t** (la touche de contrôle et la lettre 't' simultanément). Continuez à appuyer sur **CTRL+t** pour inverser et revenir à la première ligne où vous avez commencé la navigation. C'est-à-dire que vous pouvez conserver pressées **CTRL+]** et ensuite taper **CTRL+t** pour revenir. Vous pouvez refaire ceci aussi souvent que vous le désirez pour avoir une navigation complète au travers de toutes les fonctions C ou C++.

## 5.1 Ctags pour ESQL

Puisque ctags ne supporte pas directement le langage Embedded SQL/C (ESQL), le script shell suivant peut être utilisé pour créer les marques pour esql. ESQL/C est un ensemble de commandes SQL de base de donnée à l'intérieur de programmes « C ». Le ESQL/C d'Oracle est appellé Pro*C et Sybase, Informix ont ESQL/C et PostgreSQL a produit « ecpg ».

Sauvez ce fichier sous « sqltags.sh » et tapez chmod a+rx tags_gen.sh.

```
#!/bin/sh

# Programme pour créer les ctags pour les fichiers ESQL, C++ et C
ESQL_EXTN=pc
tag_file1=tags_file.1
tag_file2=tags_file.2

which_tag=ctags

rm -f $tag_file1 $tag_file2 tags

aa=`ls *.$ESQL_EXTN`
#echo $aa
for ii in $aa
do
        #echo $ii
        jj=`echo $ii | cut -d'.' -f1`
        #echo $jj

        if [ ! -f $jj.cpp ]; then
                echo " "
                echo " "
                echo "*******************************************************"
                echo "Les fichiers ESQL *.cpp files n'existent pas..."
                echo "Vous devez générer les fichiers *.cpp à partir des *.pc"
                echo "en utilisant le pré-compilateur Oracle Pro*C ou Sybase"
                echo "ou le pré-compilateur Informix esql/c."
                echo "Puis relancez cette commande"
                echo "*******************************************************"
                echo " "
                exit
        fi

        rm -f tags
```

```
        $which_tag $jj.cpp
        kk=s/$jj\.cpp/$jj\.pc/g

        #echo $kk > sed.tmp
        #sed -f sed.tmp tags >> $tag_file1

        #sed -e's/sample\.cpp/sample\.pc/g' tags >> $tag_file1
        sed -e $kk tags >> $tag_file1
done

# S'occupe des fichiers C++/C - exclut les fichiers ESQL *.cpp
rm -f tags $tag_file2
bb=`ls *.cpp *.c`
aa=`ls *.$ESQL_EXTN`
for mm in $bb
do
        ee=`echo $mm | cut -d'.' -f1`
        file_type="NOT_ESQL"
        # Exclut les fichiers ESQL *.cpp et *.c
        for nn in $aa
        do
                dd=`echo $nn | cut -d'.' -f1`
                if [ "$dd" = "$ee" ]; then
                        file_type="ESQL"
                        break
                fi
        done

        if [ "$file_type" = "ESQL" ]; then
                continue
        fi

        rm -f tags
        $which_tag $mm
        cat tags >> $tag_file2
done

mv -f $tag_file2 tags
cat  $tag_file1 >> tags
rm -f $tag_file1

# Doit sortir le fichier des marqueurs pour fonctionner correctement...
sort tags > $tag_file1
mv $tag_file1 tags
```

## 5.2 Ctags pour les programmes JavaScript, les scripts shell Korn, Bourne

Le script shell donné ci-dessous peut être utilisé pour générer les marques pour une très large variété de programmes écrits en JavaScript, les scripts PHP/FI, Korn, C, Bourne et beaucoup d'autres. C'est un module très générique.

Sauvez ce fichier sous tags_gen.sh et tapez chmod a+rx tags_gen.sh.

```
#!/bin/sh

tmp_tag=tags_file
tmp_tag2=tags_file2

echo " "
echo " "
echo " "
echo " "
echo " "
echo "Génère les marqueurs pour..."
while :
do
        echo "Entrer l'extension du fichier pour lequel vous voulez générer des marqueurs."
        echo -n "Les extensions de fichiers peuvent être sh, js, ksh, etc... : "
        read ans

        if [ "$ans" == "" ]; then
                echo " "
                echo "Mauvaise entrée. Essayez encore !"
        else
                break
        fi
done

rm -f $tmp_tag

aa=`ls *.$ans`

for ii in $aa
do
        jj=`echo $ii | cut -d'.' -f1`
        #echo $jj
        cp $ii $jj.c
        ctags $jj.c
        echo "s/$jj.c/$ii/g" > $tmp_tag2
        sed -f $tmp_tag2 tags >> $tmp_tag
        \rm -f tags $jj.c
done

sort $tmp_tag > tags

rm -f $tmp_tag $tmp_tag2
```

## 5.3 Déboguer avec gdb

Vous utiliserez gdb extensivement avec Vi. Le déboguage est l'un des plus importants aspects de la programmation en tant que coût majeur du développement et des tests des projets.

Pour déboguer des programmes C/C++ vous utiliserez l'outil « gdb ». Voyez **'man gdb'**. Vous devrez compiler vos programmes avec l'option -g3 comme

```
        gcc -g3 foo.c foo_another.c sample.c
```

Pour configurer des alias utiles :

```
  Configurez l'alias dans votre ~/.bash_profile
        alias gdb='gdb -directory=/home/src -directory=/usr/monnom/src '
  Donnera -
        gdb foo.cpp
        gdb> dir /hom2/another_src
        Ceci ajoutera un chemin à la recherche de fichier
        gdb> break 'some_class::func<TAB><TAB>
  Ce qui complètera le nom de la fonction en vous évitant le temps de
frappe... et sortira comme -
        gdb> break 'some_class::function_foo_some_where(int aa, float bb)'
```

Taper la touche TAB deux fois permet le complément de la ligne de commande, sauvant ainsi beaucoup de temps de frappe. C'est l'une des techniques les plus importantes pour l'utilisation de gdb.

Pour obtenir de l'aide en ligne -

```
        gdb> help
  Donne l'aide en ligne
        gdb> help breakpoints
  Donne plus de détails sur les points d'ancrage.
```

Pour placer les points d'ancrage et effectuer du déboguage

```
        unixprompt> gdb exe_filename
        gdb> b main
  Ceci mettra un point d'ancrage dans la fonction main()
        gdb> b 123
  Ceci mettra un point d'ancrage à la ligne 123 du fichier courant
        gdb> help breakpoints
    Donne plus de détails sur les points d'ancrage.
```

Pour analyser des core dumps :

```
        unixprompt> gdb exe_filename  core
        gdb> bt
Donne une trace de retour des fonctions et les numéros de lignes où le
programme a échoué
        gdb> help backtrace
  Donne plus de détails sur la trace de retour.
```

Vous pouvez aussi utiliser une version GUI de gdb appellée xxgdb.

Outils de perte de mémoire -

- Freeware Electric Fence sous linux cd
- Commercial tools Purify `http://www.rational.com`
- Insure++ `http://www.insure.com`

# 6 Aide de ViM en ligne

Voyez les pages de manuel en ligne. Au prompt unix, tapez **'man vim'** et **'man gvim'**.

Ou lors d'une session gvim tapez :help pour obtenir la page d'aide. Voyez aussi le 8 ()

# 7 Pages web de ViM et liens ViM

La page principale de ViM se trouve sur `http://www.vim.org` , et son site miroir aux US est sur `http://www.us.vim.org` .

La FAQ ViM est sur `http://www.grafnetix.com/~laurent/vim/faq.html`

et sur `http://www.vim.org/faq` .

La page ViM d'Eli se trouve sur `http://www.netusa.net/~eli/src/vim.html` .

La page des amoureux de Vi sur `http://www.cs.vu.nl/~tmgil/vi.html` .

Le guide de référence sur ViM sur `http://scisun.sci.ccny.cuny.edu/~olrcc/vim/` .

Les listes de diffusion ViM sont sur `http://www.findmail.com/listsaver/vimannounce.html` et `http://www.vim.org/mail.html` .

Les archives des listes sont conservées sur :

- `http://www.egroups.com/group/vim`
- `http://www.egroups.com/group/vimdev`
- `http://www.egroups.com/group/vimannounce`

Les macros ViM sont sur `http://www.grafnetix.com/~laurent/vim/macros.html` .

# 8 Tutoriel ViM

## 8.1 Tutoriels ViM sous la main

Sur les systèmes Linux, on trouve souvent le tutoriel dans /usr/doc/vim-common-5.*/tutor, sur les autres systèmes Unix cherchez le répertoire où ViM est installé et cherchez le répertoire doc.

```
bash$ cd /usr/doc/vim-common*/tutor
bash$ less README.txt
bash$ cp tutor $HOME
bash$ cd $HOME
bash$ less tutor
```

## 8.2 Tutoriels Vi sur Internet

- Purdue University `http://ecn.www.ecn.purdue.edu/ECN/Documents/VI/`
- Quick Vi tutorial `http://linuxwww.db.erau.edu/LUG/node165.html`
- Advanced Vi tutorial `http://www.yggdrasil.com/bible/bible-src/user-alpha-4/guide/node171.html`

- Tutorials `http://www.cfm.brown.edu/Unixhelp/vi_.html`
- Tutorials `http://www.linuxbox.com/~taylor/4ltrwrd/section3_4.html`
- Unix world online vi tutorial `http://www.networkcomputing.com/unixworld/unixhome.html`
- Univ of Hawaii tutorial `http://www.eng.hawaii.edu/Tutor/vi.html`
- InfoBound `http://www.infobound.com/vi.html`
- Cornell Univ `http://www.tc.cornell.edu/Edu/Tutor/Basics/vi/`
- Vi Lovers home page `http://www.cs.vu.nl/~tmgil/vi.html`
- Après Sept 2000, sera sur `http://www.thomer.com/thomer/vi/vi.html`
- Beginner's Guide to vi `http://www.cs.umr.edu/unixinfo/general/packages/viguide.html`
- vi Help file `http://www.vmunix.com/~gabor/vi.html`
- ViM FAQ `http://www.math.fu-berlin.de/~guckes/vim/faq/`

Il y a de nombreux tutoriels Vi sur Internet. Sur Yahoo (Lycos, excite ou Hotbot), entrer « Vi Tutorial » dans le champ de recherche vous renverra de nombreux pointeurs.

# 9 Tutoriel Vi

Dans ce tutoriel, nous décrirons quelques commandes et concepts **vi** avancés, vous pourrez ainsi apprécier la puissance de **vi** et décider de construire vos connaissances avec les commandes **vi**. Quasiment toutes les références listent les commandes disponibles, mais beaucoup ne montrent pas comment ces commandes interagissent ; ce point précis est le thème principal de ce tutoriel.

## 9.1 Commandes du mouvement du curseur

Les commandes du mouvement du curseur de **vi** vous permettent de positionner le curseur dans le fichier et/ou à l'écran de manière efficace, avec un nombre minimal de frappe de touches. Il y a de nombreuses commandes contrôlant les mouvements du curseur - n'essayez pas de toute les mémoriser en une fois ! Plus tard, nous verrons que la majeure partie de la puissance de **vi** vient du mélange entre les commandes de mouvement du curseur et les autres commandes pour effacer, changer, copier, et filtrer le texte.

Veuillez éditer un gros fichier texte (disons, **wknight**) afin d'expérimenter chaque commande décrite. Gardez en tête que ces commandes ne marchent qu'en Mode Commande, et pas en Mode Insertion ; si vous voyez vos « commandes » dans votre texte, appuyez sur ESC pour retourner en Mode Commande.

- **touches fléchées** : Ainsi que nous l'avons vu, les touches curseur permettent de se déplacer avec un simple caractère vers la gauche, le bas, le haut et la droite. Les mouvements au-delà du haut du fichier, en dessous du bas, à droite de la fin de la ligne, ou à gauche du début ne sont pas autorisés (pas de coupure de ligne).
- **hjkl** : Lorsque **vi** a été écrit (vers 1978), de nombreux terminaux sur systèmes UNIX n'avaient pas de touches fléchées ! **h, j, k,** et **l** ont été choisies comme commandes pour se déplacer vers la gauche, le bas, le haut, et la droite, respectivement. Essayez les ! La plupart des intégristes de **vi** les préfèrent aux touches fléchées car :
  - **(a)** elles sont à la même place sur tous les claviers, et

- **(b)** elles se placent agréablement sous les doigts, au contraire de la plupart des touches fléchées, qui sont arrangées en boîte ou en « T » ou sous une autre forme non linéaire.

Pourquoi h, j, k, et l ? Eh bien, dans le code des caractères ascii, CTRL-H est l'effacement (déplacement vers la gauche), CTRL-J le retour chariot (déplacement vers le bas), et bien entendu, k et l sont proches de h et j, et comme vous le voyez, ces touches forment une combinaison mnémonique.

- **0** : (« zéro », et pas « oh ») Déplacement au début de la ligne courante. (Pour essayer ceci et les quelques commandes suivantes, utilisez les touches du curseur ou **h j k l** pour vous déplacer vers une ligne indentée contenant quelques caractères « e ». Si vous ne pouvez trouver de ligne indentée dans votre fichier, créez en une en insérant quelques espaces au début de la ligne.)
- **^** : Déplacement sur le premier caractère non-blanc de la ligne courante (pour une ligne indentée, 0 et ^ ont des significations différentes).
- **$** : Déplacement sur le dernier caractère de la ligne courante.
- **tC** : Déplacement jusqu'au (mais pas sur) le prochain caractère C de la ligne courante (tapez 0, puis tapez te. Ceci vous déplacera vers le premier e de la ligne courante).
- **fC** : Trouve (déplacement sur) le prochain caractère C de la ligne courante (tapez fe, et le curseur trouvera - c'est-à-dire se mettra sur - le prochain e de la ligne courante).
- **TC** : Déplacement jusqu'au (mais pas sur) le précédent caractère C de la ligne courante (tapez $, puis Te).
- **FC** : Trouve (déplacement sur) le précédent caractère C de la ligne courante (tapez Fe).
- **n|** : Déplacement sur la colonne n de la ligne courante (tapez 20 | ; les chiffres 2 et 0 ne seront pas affichés lorsque vous les taperez, mais lorsque vous presserez | le curseur se déplacera en colonne 20).

  Essayez quelquez trucs avec t f T F | . Lorsque vous faîtes quelque chose d'illégal, **vi** émettra un bip.
- **w** : Déplacement au début du prochain « petit » mot (un « petit » mot consiste en une suite ininterrompue de caractères alphanumétiques ou de caractères de ponctuation, mais pas un mélange de caractères de ponctuation et alphanumériques). Essayez de taper w une douzaine de fois – notez ce qui arrive aux ponctuations.
- **W** : Déplacement au début du prochain « grand » mot (mélange alphanumérique et ponctuation). Essayez de taper W une douzaine de fois.
- **b** : Retour au début d'un « petit » mot.
- **B** : Retour au début d'un « grand » mot.
- **e** : Déplacement à la fin d'un « petit » mot.
- **E** : Déplacement à la fin d'un « grand » mot.
- **+ Return** : Déplacement sur le premier caractère non-blanc sur la même ligne (+ et la touche Entrée ont le même effet).
- **-** : Déplacement sur le premier caractère non-blanc de la ligne précédente.
- **)** : Déplacement sur la fin d'une phrase (une phrase se termine soit par une ligne blanche, ou un point ou une marque d'exclamation suivis par deux caractères d'espace ou la fin de la ligne. Un point ou une marque d'exclamation suivis par un seul caractère d'espace ne termine pas une phrase ; ceci est un comportement correct, en accord avec les règles traditionnelles de la manière dont les phrases

doivent apparaître dans les documents imprimés, mais apparaît souvent comme faux pour ceux qui n'ont jamais utilisé une classe typographique correcte.)

NdT : Ceci n'est valable qu'en typographie anglaise. En typographie française, une phrase se termine par un point (ou une marque d'exclamation) suivit par une espace.

- **(** : Déplacement au début d'une phrase.
- **}** : Déplacement à la fin d'un paragraphe (les paragraphes sont séparés par des lignes blanches, par définition avec **vi**).
- **{** : Déplacement au début d'un paragraphe.
- **H** : Déplacement vers la position première (la ligne du haut) de l'écran.
- **M** : Déplacement au milieu de la ligne à l'écran.
- **L** : Déplacement sur la dernière ligne de l'écran.
- **nG** : Déplacement sur la ligne n. Si n n'est pas donné, déplacement sur la dernière ligne du fichier (essayez 15G pour vous déplacer sur la ligne 15, par exemple. La commande CTRL-G affiche le nom du fichier, quelques informations sur l'état, et le numéro de la ligne actuelle. Pour vous déplacer au début du fichier : 1G).
- **CTRL-d** : Déplacement vers le bas d'un demi-écran (voir note).
- **CTRL-u** : Déplacement vers le haut d'un demi-écran (voir note).
- **CTRL-f** : Déplacement vers le bas d'un écran (voir note).
- **CTRL-b** : Déplacement vers le haut d'un écran (voir note).
- **Note** : Ces quatres commandes de déplacement ne peuvent être utilisées avec les commandes d'effacement, de changement, de copie ou de filtre.
- **/reg_exp** : Déplacement sur la prochaine occurence de l'expression rationnelle reg_exp. Lorsque vous tapez /, le curseur se déplace vers le coin en bas à gauche de l'écran et attend que vous tapiez l'expression rationnelle. Tapez la touche Entrée pour finir ; **vi** cherchera alors dans la suite du fichier la prochaine occurence de l'expression rationnelle. Par exemple, tapez /the puis Entrée. Ceci vous déplacera sur la prochaine apparition de the, peut-être mise en évidence au milieu d'un mot plus long (other, weather, etc.). Si vous tapez juste / puis Entrée, **vi** cherchera la prochaine apparition de la dernière expression rationnelle que vous aviez cherché.
- **n** : A le même effet que de presser / et Entrée ; c-à-d recherche la prochaine occurence de la dernière expression rationnelle que vous aviez cherché.
- **?reg_exp** : Recherche en arrière, et pas en avant. Si la reg_exp n'est pas donnée, recherche la dernière expression rationnelle entrée. Les 2 / et ? sont tournants, donc rechercher « plus bas » que le bas ou « plus haut » que le haut du fichier est légal.
- **N** : Identique à ? et Entrée.

## 9.2 Compteurs de répétitions

La plupart des commandes de mouvements présentées ci-dessus peuvent être précédées d'un compteur de répétitions ; le mouvement est simplement répété le nombre de fois donné :

- **3w** : Déplacement en avant de 3 mots.

- **5k** : Déplacement vers le haut de 4 caractères.
- **3fa** : Trouve le 3ème « a » successif de la ligne courante.
- **6+** : Descend de 6 lignes.

Pour certaines commandes, les « compteurs de répétitions » ont des significations spéciales :

- **4H** : Déplacement vers la ligne 4 de l'écran (touche home et 3).
- **8L** : Déplacement sur la 8ème ligne à partir du bas de l'écran.
- **3$** : Déplacement à la fin de la 3ème ligne plus bas.

Pour plusieurs commandes (telles que ^) le compteur de répétition est ignoré ; pour d'autres (par exemple, / et ?) il est illégal.

## 9.3 Effacer du texte

Nous avons vu que **dd** efface la ligne courante. Ceci peut être utilisé avec un compteur de répétitions : 3dd efface trois lignes, la ligne courante et les 2 lignes suivantes.

La commande d peut être utilisée comme un « préfixe » pour la plupart des commandes de mouvement ci-dessus pour effacer à peu près toute sorte de parties de texte. Lorsqu'elles sont utilisées avec d, les commandes de mouvements sont appellées des spécificateurs de cibles. On peut donner un compteur de répétition à d (lorsque vous essayez ces expériences, rappellez vous d'appuyer sur u après chaque commande pour annuler l'effacement).

- **dw** : Efface le prochain « petit » mot.
- **d3w** : Efface les 3 prochains « petits » mots.
- **3dw** : Trois fois, efface le prochain « petit » mot.
- **3d3w** : Trois fois, efface les 3 prochains « petits » mots (c'est-à-dire, efface les 9 prochains « petits » mots).
- **d+** : Efface la ligne actuelle et la ligne suivante.
- **d/the** : Efface à partir du caractère courant jusqu'à, mais sans inclure, la prochaine apparition de « the ».
- **d$** : Efface jusqu'à la fin de la ligne.
- **d0** : Efface jusqu'au début de la ligne.
- **d30G** : Efface la ligne courante jusqu'à et incluant la ligne 30.
- **dG** : Efface la ligne courante jusqu'à et incluant la dernière ligne.
- **d1G** : Efface la ligne courante jusqu'à et incluant la ligne 1.

Pour effacer de simples caractères, utilisez x. x peut être utilisé en utilisant un compteur répétitif :

- **15x** : Efface le caractère courant et les 14 suivants.

x est simplement une abbréviation de d1 ; c'est-à-dire efface un caractère à droite.

## 9.4 Changer le texte

La commande c est similaire à d, à part qu'elle change le mode de **vi** en insertion, autorisant le texte original (non désiré) à être changé en quelque chose d'autre.

Par exemple, placez le curseur sur le début d'un mot (tapez w pour arriver au début du prochain mot). Ensuite, tapez cw pour changer ce mot. À l'écran, le dernier caractère de ce mot en cours de changement sera remplacé par un symbole **$** indiquant la fin du changement ; tapez un nouveau mot (vous réécrirez le mot original à l'écran) et tapez la touche ESC lorsque vous aurez fini. Votre entrée peut être plus longue ou plus courte que le mot en cours de changement.

Placez le curseur au début d'une ligne contenant au moins trois mots, et taper c3w pour changer ces trois mots. Essayez c$ pour changer la fin de la ligne actuelle. Dans tous les cas où le changement affecte uniquement la ligne courante, la fin du changement est indiquée avec $.

Lorsqu'un changement affecte plus que la ligne courante, **vi** efface le texte original de l'écran et se place en mode insertion. Par exemple, essayez c3+ pour changer la ligne courante et les trois suivantes ; **vi** supprime les quatres lignes originales de l'écran et se place en mode d'insertion sur une nouvelle ligne blanche. Comme toujours, tapez la touche ESC lorsque vous aurez fini d'entrer votre nouveau texte.

Quelques autres commandes de changement :

- **cc** : Change la ligne courante.
- **5cc** : Change cinq lignes (courante et quatre suivantes).
- **c/the** : Changer du caractère courant jusqu'à, mais sans inclure, la prochaine occurence de « the ».
- **c$** : Change jusqu'à la fin de la ligne.
- **c30G** : Change de la ligne courante jusqu'à la ligne 30 incluse.
- **cG** : Changer de la ligne courante jusqu'à et incluant la dernière ligne.
- **c1G** : Changer la ligne courante jusqu'à la ligne 1 incluse.

## 9.5 Emmener (copier) du texte

La commande y emmène une copie du texte dans un buffer ; le texte copié peut être placé (ou collé) n'importe où dans le fichier en utilisant p ou P.

La forme la plus simple de copie est yy pour copier la ligne courante ; après yy, essayez p pour mettre une copie de la ligne copiée après le curseur. En suivant yy, vous pouvez faire autant de copie de la ligne emmenée que vous le voulez en vous déplaçant dans le fichier et en tapant p.

Pour copier plusieurs lignes, essayez, par exemple, 5yy (copie la ligne courante et les 4 lignes suivantes). p place une copie des lignes emmenées après le curseur ; la séquence 5yyp « marche » mais ce n'est probablement pas ce que vous voudriez faire. La commande P fonctionne comme p, mais place une copie de la ligne au-dessus du curseur ; essayez la séquence 5yyP.

Autres commandes de copie :

- **y3w** : Copie 3 mots.
- **y$** : Copie jusqu'à la fin de la ligne.
- **y1G** : Copie de la ligne courante jusqu'à la ligne 1 incluse.

## 9.6 Filtrer le texte

La commande de filtrage **!** demande le nom d'une commande UNIX (qui doit être un filtre), passe les lignes sélectionnées par ce filtre, en remplaçant les lignes sélectionnées dans le buffer **vi** avec la sortie de la commande filtrante. La capacité de **vi** à passer des parties arbitraires de texte au travers de tout filtre UNIX ajoute une flexibilité incroyable à **vi**, sans « coût supplémentaire » de taille ou de performance à **vi** même.

Quelques exemples peuvent aider l'illustration. Créez une ligne dans votre fichier contenant juste le mot « who » et absolument aucun autre texte. Placez le curseur sur cette ligne, et tapez **!!**. Cette commande est analogue à dd, cc, ou yy, mais au lieu d'effacer, de changer ou de copier la ligne courante, elle filtre la ligne courante. Lorsque vous pressez le second !, le curseur descend vers le coin en bas à gauche de l'écran et un simple ! est affiché, vous demandant d'entrer le nom d'un filtre. En tant que nom de filtre, tapez « sh » et pressez la touche Entrée. **sh** (le shell Bourne) est un filtre ! Il lit l'entrée standard, exécute une partie de l'entrée (c'est-à-dire qu'il exécute des commandes), et envoie sa sortie (la sortie de ces commandes) à la sortie standard. Filtrer la ligne contenant « who » au travers de « sh » remplace la ligne contenant « who » par la liste des utilisateurs du système courant – directement dans votre fichier !

Essayez de répéter ce procédé avec **date**. C'est-à-dire, créez une ligne ne contenant rien d'autre que le mot **date**, puis placez le curseur sur cette ligne, et tapez **!!sh** et la touche Entrée. La ligne contenant **date** est remplacée par la sortie de la commande **date**.

Mettez votre curseur sur la première ligne de la sortie de « who ». Comptez le nombre de lignes. Supposons, par exmple, que ce nombre soit six. Sélectionnez alors ces six lignes à filtrer au travers de sort ; tapez **6!!sort** et la touche Entrée. Les six lignes seront passées à sort, et la sortie de sort remplacera les six lignes d'origine.

La commande filtre peut uniquement être utilisée sur des lignes complètes, pas sur des caractères ou des mots.

Quelques autres commandes de filtres (ici, « CR » indique pressez Entrée) :

- **!/the CR sort CR** : Sort la ligne courante jusqu'à la ligne suivante contenant « the » incluse.
- **!1Ggrep the CR** : Remplace la ligne courante jusqu'à la ligne 1 incluse par les lignes contenant « the ».
- **!Gawk '{print $1}' CR** : De la ligne courante jusqu'à la fin du fichier, remplace chaque ligne par son premier mot.

## 9.7 Marquer des lignes et des caractères

Vous pouvez marquer des lignes et des caractères pour être utilisés en tant que cible pour des mouvements, effacement, changement, copie, et filtration en utilisant la commande mc, où c est une lettre minuscule.

Par exemple, mettez le curseur au milieu d'un mot et tapez ma. Ceci marque le caractère sous le curseur sous la marque a.

Maintenant, déplacez le curseur en dehors du caractère marqué vers une ligne différente (utilisez les flèches curseur, CTRL-u, ou autre). Pour retourner à la ligne marquée, tapez 'a (c'est-à-dire apostrophe, puis a). Ceci vous place sur le premier caractère non-blanc de la ligne contenant la marque a.

Sortez de la ligne encore une fois. Pour retourner au caractère marqué, tapez ‘a (apostrophe inverse, puis a). Ceci vous déplacera sur le caractère marqué par a.

Le marquage est habituellement utilisé avec l'effacement, le changement, la copie ou la filtration. Par exemple, déplacez le curseur sur une autre ligne que celle contenant la marque a, et pressez d'a (d, apostrophe, a). Ceci efface de la ligne courante jusqu'à la ligne marquée a incluse.

Mettez le curseur au milieu d'un autre mot et tapez mb pour mettre la marque b. Maintenant, déplacez le curseur hors de ce mot (mais seulement de quelques lignes, ainsi vous pourrez voir ce que nous allons faire plus facilement), et pressez d‘b (d, apostrophe inverse, b). Ceci efface le caractère courant jusqu'au caractère marqué par b inclus.

Comme autre exemple, pour trier la sortie de who, marquez la première ligne (ma), puis déplacez le curseur vers la dernière ligne et taper !'asort puis la touche Entrée.

Si vous sautez jusqu'à une marque et que vous décidez de revenir en arrière, de l'endroit d'où vous avez sauté, vous pouvez taper ” (reviens à la ligne) ou “ (reviens au caractère).

## 9.8 Nommer les tampons

Lorsque vous effacez, modifiez, ou copier du texte, le texte original est sauvé (jusqu'au prochain effacement, changement ou copie) dans un tampon non nommé à partir duquel il peut être mis en utilisant p ou P. En utilisant le tampon non nommé, seul le plus récent changement du texte peut être récupéré.

Si vous voulez effacer, changer ou copier plusieurs parties de texte et se souvenir de toutes (jusqu'à un maximum de 26), vous pouvez donner un nom au tampon avec la commande utilisée. Un nom de tampon est de la forme "c (double apostrophe, c minuscule).

Par exemple, tapez "ayy pour copier la ligne courante dans le tampon a, puis déplacez vous sur une autre ligne et tapez "byy pour copier cette ligne dans le tampon b. Maintenant, déplacez vous n'importe où dans le fichier et tapez "ap et "bp pour placer des copies du texte sauvé dans les tampons a et b.

Quelques autres commandes de tampon :

- **"a6yy** : Copie 6 lignes (courante et 5 suivantes) dans le tampon a.
- **"bd1G** : Efface de la ligne courante jusqu'à la ligne 1 incluse, en sauvant les lignes effacées dans le tampon b.
- **"cy'c** : Copie de la ligne courante jusqu'à la ligne marquée c dans le tampon c (les marques et les tampons sont distincts, et peuvent avoir le même nom sans que **vi** ne s'en préoccupe).

## 9.9 Substitutions

Pour échanger un bloc de texte par un autre dans les lignes de votre fichier, utilisez la commande :s. Quelques exemples de substitutions :

- **:1,$s/the/THE/g** De la ligne 1 à la dernière ligne (ligne $), remplace le texte « the » par « THE » ; le fais globalement pour toute ligne où se trouve un « the ».
- **:'a,.s/.*/ha ha/** De la ligne marquée a jusqu'à la ligne actuelle (ligne .), substitue tout ce qu'il y a sur la ligne par le texte « ha ha ».

## 9.10 Diverses « commandes double point »

Toutes les commandes de double point débutent avec « : » ; lorsque vous tapez ce symbole, le curseur saute vers le coin bas à gauche de l'écran, et un prompt à deux points est affiché, attendant que vous finissiez la commande.

Quelques exemples importants :

- **:w** Écrit le contenu du tampon dans le fichier sans quitter **vi**.

- **:w abc** Écrit le contenu du tampon dans le fichier abc (crée abc s'il n'existe pas, ou réécrit son contenu actuel s'il existe) sans quitter **vi**.
- **:1,10w abc** Écrit les lignes 1 à 10 dans le fichier abc.
- **:'a,$w abc** Écrit de la ligne marquée a jusqu'à la dernière ligne dans le fichier abc.
- **:e abc** Édite le fichier abc, au lieu du fichier actuel. **vi** affiche un message d'erreur si des changements ont été faits au fichier actuel et qui n'ont pas été sauvés avec :w.
- **:e** # Édite le fichier précédemment édité (des commandes successives :e# vont et viennent entre deux fichiers).
- **:f abc** Change le nom du fichier pour le tampon actuel vers abc.
- **:q** Quitte, à moins qu'il n'y ait des changements que vous ayez faits.
- **:q!** Quitte, en omettant tous les changements que vous pourriez avoir fait.
- **:r abc** Lit le fichier abc dans le tampon **vi** actuel, après la ligne sur laquelle se trouve le curseur (essayez :r croc pour insérer une copie du fichier croc).
- **:!cmd** Exécute la commande cmd (who, sort, ls, etc.).

## 9.11 Utiliser les options

Diverses options peuvent affecter le « confort » de **vi**. Vous pouvez afficher toutes les diverses options pouvant être utilisées en tapant set all. Vous pouvez également utiliser « :set » pour changer les options.

Par exemple, si vous désirez voir le numéro de ligne pour les lignes du fichier que vous éditez, utilisez la commande :set number. Pour supprimer l'affichage du numéro de ligne, utilisez la commande :set nonumber. La plupart des options peuvent être abrégées ; :set nu affiche le numéro des lignes et :set nonu le supprime.

Si vous utilisez :set nomagic, la signification spéciale des caractères d'expression régulière (point, astérisque, crochet, etc.) est supprimée. Utilisez :set magic pour restaurer ces significations particulières.

Quelques options ont une valeur. Par exemple, :set tabstop=4 affiche les tabulations en quatre caractères d'espace, plutôt que les huit habituels.

Si vous trouvez que vous désirez toujours certaines options placées de certaines manières, vous pouvez mettre ces commandes optionnelles dans un fichier .exrc, ou vous pouvez utiliser la variable d'environnement EXINIT pour spécifier les options désirées.

Par exemple, si votre shell par défaut est le shell Bourne, cette ligne peut aller dans votre fichier .profile :

```
EXINIT='set nomagic nu tabstop=4'; export EXINIT
```

Si votre shell par défaut est un C shell, cette ligne peut aller dans votre fichier .login :

```
setenv EXINIT 'set nomagic nu tabstop=4'
```

## 9.12 Cartographie des touches

Si vous vous apercevez que vous utilisez encore et encore la même série de commandes simples, vous pouvez les lier à une touche de commande inutilisée en utilisant la commande :map. Si votre lien inclut des caractères de contrôle comme la touche Entrée (ctrl-M en ascii) ou ESC (ctrl-[ en ascii), vous pouvez les faire précéder de ctrl-v pour supprimer leur signification classique.

Par exemple, cette commande relit ctrl-A pour déplacer le curseur de 55 lignes, puis revenir à la ligne vide la plus récente, changer ensuite cette ligne blanche par un saut de page (ctrl-L) et trois lignes blanches. C'est à dire que chaque ctrl-A paginera la page suivante, sans couper de paragraphes entre les pages.

Note : Dans cette commande, chaque caractère de contrôle est représenté par ˆC, où C est une lettre majuscule quelconque. Par exemple, ctrl-M est représenté par ˆM. De même, lorsque vous entrez cette commande vous ne verrez pas les caractères ctrl-v : chaque ctrl-v se voit remplacé par la signification spéciale du caractère de contrôle suivant, ainsi lorsque vous utiliserez la séquence ˆVˆM, tout ce que vous verrez à l'écran est un ˆM. Dans cette commande, ˆM est la touche Entrée et ˆ[ la touche ESC.

```
:map ^A  55+?^$^V^Mcc^V^L^V^M^V^M^V^M^V^[
```

## 9.13 Éditer plusieurs fichiers

Vous pouvez éditer plusieurs fichiers avec **vi** en lui donnant plusieurs noms de fichiers en arguments de ligne de commande :

```
vi croc fatherw  wknight
```

Trois commandes sont utilisées pour se déplacer entre plusieurs fichiers :

- **:n** Déplace dans le prochain fichier de la liste d'arguments (vous devez sauver les changements avec :w ou **vi** affichera un message d'erreur).
- **:N** Déplace dans le fichier précédent de la liste d'arguments (vous devez sauver les changements avec :w ou **vi** affichera un message d'erreur).
- **:rew** Rembobine et repart du premier fichier de la liste d'arguments.

Les commandes :n, :N et :rew sont quelques peu étranges, mais elles ont des bénéfices importants : le contenu des tampons ("a, "b, "c, etc.) sont connus entre les fichiers, et vous pouvez ainsi utiliser :n et :rew avec p et P pour copier le texte vers l'un ou l'autre des fichiers. De même, la plus récente chaîne de recherche des commandes / et ? sont les mêmes suivant les fichiers, et vous pouvez ainsi faire des recherches répétées sur plusieurs fichiers assez facilement.

Par exemple, tentez l'expérience suivante : tout d'abord sortez de **vi**, et exécutez ensuite **vi** avec croc et wknight en arguments :

```
$ vi croc wknight
```

Dans croc, cherchez

**/the < CR >**

Copiez cette ligne dans le tampon a :

**"ayy**

Maintenant déplacez vous sur le prochain fichier (vous n'avez fait aucun changement à croc, donc ça marchera) :

**:n < CR >**

Cherchez la ligne suivante contenant « the », sans retaper la chaîne de recherche :

**n**

Mettez une copie du tampon après la ligne courante dans wknight :

**"ap**

Descendez de deux lignes, et copiez la ligne courante dans le tampon b :

**jj"byy**

Sauvez les changements de wknight

**:w < CR >**

Maintenant, retournez à croc

**:rew < CR >**

Cherchez encore, et mettez une copie du tampon b après la ligne trouvée :

**n"bp**

Sauvez les changements, et quittez **vi**

**ZZ**

### 9.14 Remarques finales

Ce tutoriel était prévu pour vous introduire quelques unes des possibilités de **vi** que vous pouvez également retrouver dans le manuel **vi** de votre système ou qui ne sont pas mentionnées dans ce manuel (de nombreux systèmes ont des manuels de qualité très variée).

Vous ne serez pas un expert **vi** après la lecture de ce tutoriel, mais vous aurez une bonne appréciation des possibilités de **vi**. Seul le temps et l'effort peuvent vous faire devenir un expert **vi**. Mais l'efficacité et l'universalité de **vi** rend cet effort payant dans le long terme.

Vous pouvez avoir décidé que vous détestez **vi**. Très bien ! Mais prenez garde, **vi** demeure l'éditeur de texte standard sous UNIX - le seul éditeur sur lequel vous pouvez compter pour être disponible sous tout système UNIX que vous utiliserez - donc même si vous préférez utiliser quelque chose d'autre dans la vie courante, vous seriez avisé de connaître le minimum à propos de **vi** qui est couvert dans ce tutoriel.

## 10 Carte de référence ViM

### 10.1 États Vi

Vi a 3 modes :

1. ***mode commande*** - État normal et initial ; les autres reviennent ici (utilisez **ESC** pour stopper une commande partiellement tapée).
2. ***mode d'entrée*** - Atteint par différentes commandes **a i A I o O c C s S R** et terminé par **ESC** ou anomalement par interruption.

3. ***mode ligne*** - C-à-d attendant une entrée après une commande **:** , / , **?** ou un **!** (terminé par **CR**, stoppé par **CTRL-c**). **CTRL** est la touche de contrôle : **CTRL-c** signifie « control c ».

## 10.2 Commandes Shell

1. **TERM=** *code* Place le nom de code de votre terminal dans la variable **TERM**.
2. **export TERM** Transporte la valeur de **TERM** (le code du terminal) vers tout programme UNIX dépendant du terminal.
3. **tput init** Initialise le terminal pour qu'il fonctionne proprement avec divers programmes UNIX.
4. **vi** *fichier* Accède à l'éditeur **vi** afin de pouvoir éditer le fichier spécifié.
5. **vi** *fichier1 fichier2 fichier3* Rentre trois fichiers dans le tampon **vi** à éditer. Ces fichiers sont *fichier1*, *fichier2*, et *fichier3*.
6. **view** *fichier* Invoque l'éditeur vi sur le *fichier* en mode lecture.
7. **vi -R** *fichier* Invoque l'éditeur vi sur le *fichier* en mode lecture.
8. **vi -r** *fichier* Récupère le *fichier* et les éditions récentes après le crash du système.

## 10.3 Activer les options

1. **:set** *option* Active l'*option*.
2. **:set** *option=valeur* Assigne la *value* à l'*option*.
3. **:set no** *option* Désactive l'*option*.
4. **:set** Affiche les options activées par l'utilisateur.
5. **:set all** Affiche la liste de toutes les options, à la fois les options par défaut et celles activées par l'utilisateur.
6. **:set** *option*? Affiche les valeurs de l'*option*.

## 10.4 Notations utilisées

Notations :

1. **CTRL-c CTRL** est la touche de contrôle : **CTRL-c** signifie « control c » ;
2. **CR** est le retour chariot (touche Entrée).

## 10.5 Interrompre, annuler

- **ESC** Arrête l'insertion ou une commande incomplète ;
- **CTRL-? CTRL** est la touche de contrôle : **CTRL-?** signifie « control ? » supprime ou interrompt des interruptions ;
- **CTRL-l** réaffiche/rafraîchit l'écran si ctrl-? l'a malmené.

## 10.6 Manipulation de fichier

- **ZZ** Sauve le fichier et sort de vi.
- **:wq** Sauve le fichier et sort de vi.
- **:w** Écrit le fichier courant.
- **:w!** Force l'écriture du fichier courant, si le fichier est en lecture seule.
- **:w*nom*** Écrit dans le fichier *nom*.
- **:q** Sort de vi.
- **:q!** Force la sortie de vi (annule les changements).
- **:e name** Édite le fichier *nom*.
- **:e!** Réédite, en annulant les changements.
- **:e + name** Édite le fichier *nom*, en partant de la fin.
- **:e + n** Édite en partant de la ligne *n*.
- **:e #** Édite un fichier alternatif.
- **:n** Édite le fichier suivant de la *liste des arguments*.
- **:args** Liste les fichiers de la liste actuelle.
- **:rew** Rembobine la liste des fichiers et édite le premier.
- **:n args** Spécifie une nouvelle liste des arguments.
- **:f** Affiche le fichier actuel et la ligne.
- **CTRL-G** Synonyme de :f, affiche le fichier actuel et la ligne.
- **:ta tag** Pour marquer l'entrée du fichier par *tag*.
- **CTRL-]** :ta, le mot suivant est tag.

## 10.7 Mouvement

- **Flèches** Déplace le curseur.
- **CTRL-d** Descend d'une demi-page.
- **CTRL-u** Monte d'une demi page.
- **CTRL-f** Descend d'une page entière.
- **CTRL-b** Monte d'une page entière.
- **:0** Déplacement au début du fichier.
- **:n** Déplacement à la ligne numéro n.
- **:$** Déplacement à la fin du fichier.
- **0** Déplacement au début de la ligne.
- **^** Déplacement au premier caractère non-blanc.

- **$** Déplacement au début de la ligne.
- **CR** Déplacement au début de la ligne suivante.
- **-** Déplacement au début de la ligne précédente.
- **%** Trouve le crochet correspondant.
- **G** Va à la ligne (défaut à la dernière ligne).
- **]]** section/fonction suivante.
- **[[** section/fonction précédente.

## 10.8 Positionnement en ligne

- **H** Première ligne de l'écran.
- **L** Dernière ligne de l'écran.
- **M** Ligne du milieu de l'écran.
- **+** Ligne suivante, sur le premier non-blanc.
- **-** Ligne précédente, sur le premier non-blanc.
- **CR** Entrée, pareil que +.
- **j** Ligne suivante, même colonne.
- **k** Ligne précédente, même colonne.

## 10.9 Positionnement des caractères

- **0** Début de la ligne.
- **$** Fin de la ligne.
- **h** Avance.
- **l** Recule.
- **SPACE** Pareil que l.
- **fx** Trouve x en avant.
- **Fx** Trouve x en arrière.
- **;** Répète le dernier f F.
- **,** Inverse de ; .
- **|** Vers la colonne spécifiée.
- **%** Trouve le { ou } correspondant.

## 10.10 Mots, phrases, paragraphes

- **w** Mot en avant.
- **b** Mot en arrière.
- **e** Fin du mot.
- ) Phrase suivante.
- ( Phrase précédente.
- } Paragraphe suivant.
- { Paragraphe précédent.
- **W** Mot délimité par du blanc.
- **B** Retour d'un mot blanc.
- **E** Fin d'un mot blanc.

## 10.11 Marquage et retour

- **‘‘** (taper deux fois la touche ‘) Contexte précédent.
- **’’** (taper deux fois la touche ’) Contexte précédent au premier non-blanc de la ligne.
- **mx** Marque la position avec la lettre x.
- **‘x** (touche ‘ et lettre x) Va à la marque x.
- **’x** Va à la marque x au premier non-blanc de la ligne.

## 10.12 Corrections au cours de l'insertion

- **CTRL-h** Efface le dernier caractère.
- **CTRL-w** Efface le dernier mot.
- **erase** Taper la touche DELETE, pareil que CTRL-h.
- **kill** Votre touche kill, efface l'entrée de la ligne.
- \ Échappement de CTRL-h, efface et kill.
- **ESC** Stoppe l'insertion, retourne en commande.
- **CTRL-?** Interrompt, termine l'insertion.
- **CTRL-d** Retour de tabulation sur un *autoindent*.
- **CTRL-v** Affiche un caractère non affichable.

## 10.13 Ajuster l'écran

- **CTRL-l** Efface et redessine.
- **CTRL-r** Redessine, élimine les lignes @.
- **z-CR** Redessine avec la ligne courante en haut.
- **z-** Redessine, avec la ligne courante au bas de la fenêtre.
- **z.** Redessine, avec la ligne courante au centre de la fenêtre.
- **/pat/z-** Ligne contenant *pat* en bas.
- **tn** Utilise une fenêtre de n lignes.
- **CTRL-e** Descend la fenêtre d'une ligne.
- **CTRL-y** Remonte la fenêtre d'une ligne.

## 10.14 Effacer

- **x** Efface le caractère sous le curseur.
- **X** Efface le caractère avant le curseur.
- **D** Efface jusqu'à la fin de la ligne.
- **d^** Efface jusqu'au début de la ligne.
- **dd** Efface la ligne.
- **ndd** Efface *n* lignes en commençant à la ligne actuelle.
- **dnw** Efface *n* mots en partant du curseur.

## 10.15 Insérer, changer

- **i** Passe en mode d'insertion avant le curseur.
- **I** Passe en mode d'insertion avant le premier caractère non-blanc.
- **a** Passe en mode d'insertion après le curseur.
- **A** Passe en mode d'insertion après la fin de la ligne.
- **o** Ouvre une nouvelle ligne en dessous de la ligne courante et passe en mode d'insertion.
- **O** Ouvre une nouvelle ligne au dessus de la ligne courante et passe en mode d'insertion.
- **r** Remplace le caractère en dessous du curseur (ne passe PAS en mode d'insertion).
- **R** Entre en mode d'insertion en remplaçant les caractères.
- **C** shift-c. Change le reste de la ligne.
- **D** shift-d. Efface le reste de la ligne.
- **s** Substitue des caractères.
- **S** Substitue des lignes.
- **J** Joint les lignes.

## 10.16 Copier et coller

Le tampon de copie est rempli par *TOUTE* commande d'effacement, ou explicitement par **Y** et **yy**.

- **Y** Copie la ligne courante dans le tampon.
- ***n*yy** Copie *n* lignes en partant de la ligne actuelle dans le tampon.
- **p** Colle le tampon après le curseur (ou sous la ligne actuelle).
- **P** Colle le tampon avant le curseur (ou avant la ligne actuelle).
- **"*x*p** Colle à partir du buffer x.
- **"*x*y** Copie dans le tampon x.
- **"*x*d** Efface dans le tampon x.

## 10.17 Opérateurs (utiliser des doubles pour agir sur les lignes complètes)

- **d** Effacer.
- **c** Changer.
- < Déplacement gauche.
- > Déplacement droit.
- **!** Filtrer au travers de la commande.
- = Indenter pour LISP.
- **y** Copier le texte dans le tampon.

## 10.18 Chercher et remplacer

- /***texte*** Cherche en avant pour *texte*.
- ?***texte*** Cherche en arrière pour *texte*.
- **n** Répète la dernière recherche dans la même direction.
- **N** Répète la dernière recherche dans la direction inverse.
- / Répète la dernière recherche en avant.
- **?** Répète la dernière recherche en arrière.
- **[ addr ] s/from/to/ [ g ]** Cherche une occurence de *from* et remplace par ***to*** dans la ligne actuelle, ou dans le bloc. ***addr*** (deux numéros de lignes séparés par une commande ; 1,$ est le fichier entier) remplace une occurence par ligne, ou toutes les occurences si ***g*** est spécifié. Par exemple, :3,20s/unmot/unautre/g remplacera "unmot" par "unautre" de la ligne 3 à la ligne 20. 'g' est global et signifie que toutes les occurences de "unmot" seront remplacées.

### 10.19 Général

- **:sh** Donne un shell (à quitter avec CTRL-d).
- **:!*commande*** Lance un shell pour exécuter *commande*.
- **:set number** Active la numérotation des lignes.
- **:set nonumber** Désactive la numérotation des lignes.

### 10.20 Commandes d'édition de ligne

- **:** Prévient **vi** que les prochaines commandes seront des commandes pour l'éditeur de ligne.
- **:sh** Retourne temporairement au shell pour exécuter quelques commandes shell sans quitter **vi**.
- **CTRL-d** Sort du shell temporaire et retourne sous **vi** afin de pouvoir éditer la fenêtre actuelle.
- **:***n* Va à la *n*ème ligne du tampon.
- **:***x,z***w** *fichier* Écrit les lignes *x* à *z* dans un nouveau fichier appellé *fichier*.
- **:$** Déplace le curseur au début de la dernière ligne du tampon.
- **:.,$d** Efface toutes les lignes de la ligne actuelle à la dernière ligne.
- **:r** *fichier* Insère le contenu du fichier *fichier* sous la ligne actuelle du tampon.
- **:s***/texte/autre_texte/* Remplace la première apparition de *texte* sur la ligne actuelle par *autre_texte*.
- **:s***/texte/autre_texte/g* Remplace toutes les apparitions de *texte* sur la ligne actuelle par *autre_texte*.
- **:g***/texte/s//autre_texte/g* Change toutes les apparitions de *texte* dans le tampon par *autre_texte*.

### 10.21 Autres commandes

- **u** Annule le dernier changement.
- **U** Restaure la ligne actuelle.
- **~** Change la casse.
- **J** Joint la ligne actuelle et la ligne suivante.
- **.** Répète la dernière commande de changement de texte.
- **CTRL-g** Montre le nom du fichier et le nombre de lignes.

## 11 URLs connexes

Les URLs connexes à ViM sont sur :

- C et C++ Beautifer `http://www.metalab.unc.edu/LDP/HOWTO/C-C++Beautifier-HOWTO.html`
- Linux goodies `http://www.aldev.8m.com` ou sur `http://www.aldev.webjump.com`

# 12 Autres formats de ce document

Ce document est publié sous 11 formats différents, nommément - DVI, Postscript, Latex, Adobe Acrobat PDF, LyX, GNU-info, HTML, RTF(Rich Text Format), Plain-text, pages man Unix et SGML.

- Vous pouvez obtenir ce document howto sous la forme d'une archive tar en html, dvi, postscript et sgml de :

  `ftp://metalab.unc.edu/pub/Linux/docs/HOWTO/other-formats/`

- Le format texte plein est sur :

  `ftp://metalab.unc.edu/pub/Linux/docs/HOWTO`

- Les traductions dans d'autres langages comme français, allemand, espagnol, chinois, japonais sont sur

  `ftp://metalab.unc.edu/pub/Linux/docs/HOWTO` . Toute aide de votre part pour traduire ce document dans d'autres langages est la bienvenue.

Ce document est écrit en utilisant un outil nommé « SGML tools » qui peut être obtenu de :

`http://www.sgmltools.org` . Pour compiler le source vous obtiendrez les commandes suivantes comme :

- sgml2html Vim-howto.sgml (pour générer un fichier html)
- sgml2rtf Vim-howto.sgml (pour générer un fichier RTF)
- sgml2latex Vim-howto.sgml (pour générer un fichier latex)

Les documents LaTeX peuvent être convertis en fichiers PDF en produisant simplement une sortie Postscript en utilisant **sgml2latex** (et dvips) et en utilisant la sortie via la commande Acrobat **distill** ( `http://www.adobe.com` ) comme suit :

```
bash$ man sgml2latex
bash$ sgml2latex filename.sgml
bash$ man dvips
bash$ dvips -o filename.ps filename.dvi
bash$ distill filename.ps
bash$ man ghostscript
bash$ man ps2pdf
bash$ ps2pdf input.ps output.pdf
bash$ acroread output.pdf &
```

Ou vous pouvez utiliser la commande Ghostscript **ps2pdf**. ps2pdf est un clone pour la majorité des fonctionnalités du produit Adobe's Acrobat Distiller : il convertit les fichiers PostScript en fichiers Portable Document Format (PDF). **ps2pdf** est implémenté sous la forme d'un fichier script de commandes très petit qui invoque Ghostscript, en sélectionnant un périphérique de sortie spécial nommé **pdfwrite**. Afin d'utiliser ps2pdf, le périphérique pdfwrite doit être inclus dans le makefile lors de la compilation Ghostscript ; voyez la documentation sur la compilation de Ghostscript pour les détails.

Ce document se trouve sur :

- `http://metalab.unc.edu/LDP/HOWTO/ViM-HOWTO.html`

Vous pouvez aussi trouver ce document sur les sites miroirs suivants :

- http://www.caldera.com/LDP/HOWTO/ViM-HOWTO.html
- http://www.WGS.com/LDP/HOWTO/ViM-HOWTO.html
- http://www.cc.gatech.edu/linux/LDP/HOWTO/ViM-HOWTO.html
- http://www.redhat.com/linux-info/ldp/HOWTO/ViM-HOWTO.html
- D'autres sites miroirs près de vous (à l'échelle du réseau) peuvent se trouver sur

  http://metalab.unc.edu/LDP/mirrors.html , sélectionnez un site et allez voir le fichier /LDP/HOWTO/Vim-HOWTO.html.

Afin de voir un document au format dvi, utilisez le programme xdvi. Le programme xdvi se trouve dans le paquetage tetex-xdvi*.rpm de la Redhat Linux qui peut se trouver dans ControlPanel | Applications | Publishing | TeX.

Pour lire un document dvi utilisez la commande :

```
xdvi -geometry 80x90 howto.dvi
man xdvi
```

Et redimmensionnez la fenêtre avec une souris. Voyez la page man de xdvi. Pour naviguer utilisez les flèches, les touches page up, down, ou également les lettres 'f', 'd', 'u', 'c', 'l', 'r', 'p', 'n' pour monter, descendre, centrer, page suivante, page précédente, etc. Pour supprimer le menu expert appuyez sur 'x'.

Vous pouvez lire le fichier postscript avec le programme 'gv' (ghostview) ou 'ghostscript'. Le programme ghostscript est dans le paquetage ghostscript*.rpm et le programme gv dans gv*.rpm, qui se trouvent sous ControlPanel | Applications | Graphics. Le programme gv est beaucoup plus agréable à utiliser que ghostscript.

Ghostscript et gv sont aussi disponibles sous d'autres plateformes comme OS/2, Windows 95 et NT. Vous pouvez donc lire ce document sur toutes ces plateformes.

- Prenez ghostscript pour Windows 95, OS/2, et tous les OS sur

  http://www.cs.wisc.edu/~ghost

Pour lire le document postscript utilisez la commande :

```
gv howto.ps
ghostscript howto.ps
```

Vous pouvez lire le document en html en utilisant Netscape Navigator, Microsoft Internet explorer, Redhat Baron ou tout autre des 10 navigateurs web.

Vous pouvez lire la sortie LaTeX ou LyX en utilisant LyX ou vim.

# 13 Notice de Copyright