Introduction simple à Emacs

Patrice Karatchenzeff <p.karatchentzeff@free.fr>

Version 1.1

Résumé

Ce guide est destiné à un public complètement ignorant du maniement d'un éditeur de texte complexe. Il s'adresse en mot simple et compréhensible par tous et permet de découvrir étape par étape le fonctionnement d'un éditeur moderne et très puissant. Le but de ce document n'est pas de faire une documentation complète mais un outil de prise en main rapide. Les documentations nombreuses du réseau ou en ligne (voir la partie : VI. Documentation) compléteront cette rapide formation. Si par mégarde, j'avais laissé des points obscurs ou des explications litigieuses/fausses ou bien incomplètes ou mal documentées (pour le débutant), n'hésitez pas à m'envoyer un courriel avec un petit message : « je n'ai pas compris… » ou « j'ai aimé… » pour que les nouvelles versions de ce document soient tout simplement plus conformes à vos désirs. N'hésitez pas à me signaler les fautes, quelles qu'elles soient, y compris d'ortheaugrafffe… Personne n'est parfait !

Copyright

Table des matières

Chapitre 1

Introduction

`Emacs` tient un rôle tout particulier dans le monde du logiciel libre car il est un peu leur père. Richard Stallman, fondateur de la « Free Software Fundation » (voir en annexe le texte sur la *« General Public Licence »*) en est aussi l'auteur original et le coordinateur principal de la version `GNU Emacs`.

Pourquoi `Emacs` est–il si populaire auprès des programmeurs ? Parce qu'`Emacs` est unique en son genre. J'ai lu une fois dans les groupes de discussion quelqu'un l'appeler « environnement de textes ». Hé oui, `Emacs` est plus qu'un simple éditeur de textes (tâche dans laquelle il excelle pourtant !). `Emacs` peut faire bien d'autre chose encore : lire un courrier électronique, des groupes de discussions, lancer une commande shell, lire une documentation (`man`, `info`), rapatrier des fichiers via *ftp* et aussi naviguer sur la Toile. . . `Emacs` ne fera pas votre café mais on chuchote que le projet est à l'étude :–)

Vous aurez dès–lors compris la raison de la popularité d'`Emacs` : son incroyable polyvalence en fait un outil extraordinaire et sa maîtrise un gain de productivité non négligeable.

Chapitre 2

Mode fonctionnement

`Emacs` est un logiciel écrit principalement en `LISP`, un langage interprété. Cela permet d'ajouter facilement une fonctionnalité supplémentaire à `Emacs`, même si vos connaissances en programmation sont faibles.

Comment fonctionne `Emacs` ? Il va vous falloir penser autrement et apprendre un peu. Comme tout ce qui est nouveau, cela est un peu déroutant au début, surtout si l'on se réfère à un logiciel du même type que l'on utilisait auparavant. Videz–vous l'esprit, relaxez–vous et respirez un grand coup : tout apprentissage débute par une phase d'initiation un peu pénible…

Commençons par le vocabulaire : au moins, on saura de quoi on parle ;–).

2.1 Les tampons.

`Emacs` travaille dans des tampons (*« buffer »* en anglais). Ainsi, on peut éditer un texte dans un tampon et un autre document dans un autre tampon. Il n'y a pas de limitation, autre que celle de la machine bien sûr (la limitation est ici celle de la mémoire).

2.2 le mode de fonctionnement ou état du tampon.

À chaque fois que l'on ouvre un tampon, on se trouve dans un « mode de fonctionnement ». Par défaut, le mode est le fondamental (« *fundamental* » en anglais). Un mode de fonctionnement caractérise son environnement. Par exemple, certains langages de programmation usent et abusent de parenthèses et l'on finit par s'y perdre (comme pour… le `LISP` par exemple). Le mode de fonctionnement caractéristique du `LISP` permet d'y mettre bon ordre automatiquement. Pourquoi se fatiguer quand la machine peut le faire pour vous ? Il existe ainsi des dizaines de modes de fonctionnement, chacun caractérisant un langage de programmation (`C`, `Perl`, Shell, *etc.*), un état (fondamental, texte, *etc.*) ou un langage de formatage (`TeX`, `HTML`, `sgml`, *etc.*).

2.3 les commandes.

On arrive à un sujet qui fait peur : pourquoi taper des commandes dans un éditeur de texte : je veux taper du texte et je ne sais pas programmer ! ! ! En fait, il ne faut pas s'effrayer et garder à l'esprit qu'`Emacs` est écrit en `LISP`. Toute action EST une commande sous `Emacs`, même lorsque je tape un caractère. Simplement, dans ce cas, on ne s'en rend pas compte :–)

Les commandes d'`Emacs` sont très nombreuses et sont toutes peu ou prou de la même forme : on appelle le programme `LISP` qui fait l'action que l'on veut faire. Par exemple, si je veux ouvrir (ou trouver) un fichier, je tape :

```
M-x find-file
```

puis le nom de mon fichier. Toute personne comprenant un tant soit peu l'anglais aura immédiatement traduit la seconde partie *« find–file »* par « ouvrir–fichier ». C'est donc simple : il suffit (presque :–)) de connaître l'anglais pour pouvoir se servir de `Emacs`. C'est hélas ainsi en informatique : tout est fait pour les anglophones...

Par contre, qu'est–ce que c'est que cette bizarrerie « `M-x` » ? Il faut faire là un effort d'attention. Puisque l'on peut taper des commandes sous `Emacs`, il faut que `Emacs` comprenne et fasse la part des choses entre le texte proprement dit d'une part et les commandes d'autre part. Imaginez la salade sinon pour nos pauvres amis anglophones :–). Ainsi, il a été codé certaines séquences de touches pour dire à `Emacs` : « à partir de là, et jusqu'au prochain retour chariot (la touche « Entrée » de votre clavier), tout ce qui est écrit doit être interprété comme une commande `Emacs` et être exécuté ».

D'accord, mais qu'est–ce donc que cette séquence « `M-x` » et comment la taper au clavier ? Cette séquence est une convention typographique de l'environnement `Emacs`, pour raccourcir son écriture. Il existe deux séquences principales sous `Emacs` :

`C-x` veut dire :

```
appuyer sur la touche «CONTROL» et sur «x» en même temps.
```

`M-x` veut dire :

```
appuyer sur la touche «Méta» et sur «x» en même temps.
```

La touche Méta est une survivance de l'antiquité informatique. Sur les claviers modernes, elle est remplacé par la touche « ALT ». On peut aussi utiliser la touche « Échap » mais à ce moment, il faut relâcher la touche « Échap » avant de taper le caractère suivant. Question de choix et d'habitude...

Cela parait un peu difficile au début mais on s'y fait vite. Choisissez–vous une touche Méta (« ALT » ou « Échap ») et n'en changez plus. Les automatismes devraient venir rapidement.

Tout cela est très drôle mais s'il faut taper 25 caractères à chaque fois que l'on désire faire une action, où est le gain de productivité ? Certes, dans un premier temps, on tâtonne un peu et l'on perd du temps mais `Emacs`, qui a toujours une solution en réserve, vous réserve deux bonnes surprises :

- Le *complètement* (« *completion* » en bon anglais) : comme tout shell digne de ce nom, il n'est pas nécessaire de taper toute la commande : `Emacs` la finira pour vous. Vous commencez à la taper, puis appuyer sur la touche de complètement (en l'occurence la touche de tabulation) et `Emacs` la finit pour vous. S'il y a plusieurs commandes commençant par les mêmes lettres (cas fréquent), `Emacs` les proposent toutes et il suffit de compléter quelques caractères pour finir la commande. Cela résout aussi les problèmes de mémoire car il arrive souvent que l'on se rappelle le début de la commande : l'affichage permet de finir sur la bonne commande :–). Il en va de même pour les noms de fichiers...
- Les raccourcis « clavier » (j'entends des soupirs de soulagements... ne vous détendez pas trop : la définition de raccourcis sous `Emacs` n'est peut–être pas la même que la vôtre...). Certaines commandes sont très fréquentes (comme l'ouverture d'un fichier par exemple) et `Emacs` leur a dédié une séquence de touches particulières. Ainsi, dans le cas de l'ouverture d'un fichier, au lieu de taper « `C-x find-file` », on tape tout simplement « `C-x C-f` ». [1]
- La mémorisation des commandes : si vous avez déjà tapé une commande et que vous désirez la rappeler, `Emacs` conserve un historique de tout ce qui a été tapé et permet de le rappeler facilement avec les flèches « haut » et « bas » du clavier.
  Tout cela vous parait étrangement familier en tant qu'utilisateur d'un shell moderne (`bash`, `tcsh`, `ksh`, *etc.*) ? Et oui, tous ces shells héritent des avancées apportées par `Emacs`. Comme quoi, un simple éditeur...
- Les raccourcis claviers « universels ». J'ai hésité à mettre ce paragraphe ici. Une des raisons de choisir `Emacs` comme éditeur est de vous faire gagner du temps dans un tas d'applications sous Unix car les–dîtes applications suivent généralement les séquences de touches de `Emacs` pour les déplacements : ceci est devenu un véritable standard sous Unix. Les connaître déjà est un gain de temps précieux...

2.4 l'apparence

`Emacs` se présente toujours de la forme suivante : on peut diviser son espace en trois parties. La première, la partie supérieure, contient les menus (pratique quand on a oublié le nom d'une commande par exemple). La seconde, de loin celle qui occupe le plus de place est la partie centrale où se loge le tampon proprement dit. C'est l'espace de frappe du texte en lui–même.

La troisième partie est celle du bas, en fait une seule ligne, destinée aux commandes. Dès que vous commencerez à taper une commande, vous la verrez s'inscrire dans cette partie. Certaines commandes requièrent des paramètres (comme pour la recherche d'expressions) et c'est dans cette partie qu'ils seront tapés. Enfin, les messages d'erreurs divers ainsi que les résultats des commandes y seront affichés.

Certaines commandes ont besoin de beaucoup plus d'une ligne pour s'exprimer. Par exemple, lorsque `Emacs` propose l'ensemble des fichiers disponibles du répertoire lors d'une tabulation, ceux–ci se comptent souvent par dizaines. `Emacs` subdivise alors la fenêtre du tampon en deux parties et permet le choix du fichier avec le curseur ou la souris (bouton central). On peut aussi le faire avec le clavier

[1] Cela ne parait pas simple de taper quatre touches pour une opération aussi courante que l'ouverture d'un fichier sous un éditeur de texte ! Premièrement, si vous vous y prenez bien, vous ne taperez au plus que trois touches (essayez si vous êtes droitier (et inversez dans le cas contraire) un doigt de la main gauche sur la touche « CTRL » puis en laissant le doigt gauche enfoncé un doigt de la main droite qui tape les lettres « x » et « f ». Ensuite, il faudra vous y faire : `Emacs` possède tellement de fonctions qu'il n'est pas possible de les coder en raccourci sur un caractère. Vous avez certainement devant vous un clavier à 105 touches (environ...) or `Emacs` possèdent des centaines et des centaines de fonctions. La seule possibilité est donc la combinaison de touches pour raccourcir les noms de fonctions. Mais ne vous effrayez pas : cela devient vite un état second et l'on ne s'en rend même plus compte à la fin.

en tapant `M-v` et se déplacer dans le tampon jusqu'au fichier ou la commande désirée. Il suffit alors de taper sur « Entrée » pour valider. Lorsque le choix est fait, la partie concernant le choix des fichiers disparaît automatiquement. Si par mégarde, due à une mauvaise manipulation ou mauvaise installation, la fenêtre de choix ou d'erreurs ne disparaissait pas, faîtes–le vous même.

Pour cela, rendez la fenêtre que vous désirez garder active, c'est–à–dire avec le curseur dedans. Pour déplacer le curseur d'une partie à une autre, cliquez avec le bouton gauche de la sortie ou tout simplement tapez la commande suivante :

```
C-x o
```

qui fait passer automatiquement le curseur d'une partie à une autre. Lorsque le curseur est positionné sur la partie que vous désirez garder, tapez sur :

```
C-x 1
```

et la partie occupe alors toute la hauteur normale du tampon (voir le paragraphe sur les *éditions multiples et tampons*)

Enfin, il existe une quatrième partie qui comporte différentes informations sur l'état du tampon courant. Il s'agit d'une ligne non accessible entre la partie du tampon et celle de la ligne de commande. Cette ligne est configurable pour que les informations qui y paraissent soient paramétrables. Par défaut, il apparaît le nom du tampon en édition courante, l'état du mode (« *Fundamental* » par défaut) et le numéro de la ligne courante ou sa position en pourcentage dans le texte suivant les versions de `Emacs`.

Certaines versions graphiques de `Emacs` proposent une série d'icônes entre les menus et le tampon, icônes permettant de faire des raccourcis à la souris de certaines fonctions courantes de `Emacs`. Vous verrez qu'à l'usage, ce n'est pas pratique et que l'on a vite fait de retirer cette barre d'icônes qui prend de la place inutilement.

Chapitre 3

Lancement de `Emacs`.

3.1 Quel « `Emacs` » utiliser ?

Pour tout simplifier, il y a une multitude d'implémentations de `Emacs`. Pas de chance ? En fait, si. Cela permet de choisir celle qui vous correspond le mieux... après les avoir un peu toutes essayées :–). Pour faire simple, il y a deux « `Emacs` » qui sortent du lot : le « `GNU Emacs` » et le « `XEmacs` ». Le développement de ces deux logiciels se fait en parallèle et ils possèdent pratiquement les mêmes fonctionnalités. Après c'est une histoire de guerre de religions dont sont très friands les partisans des logiciels libres :–)

Pour ne pas vous influencer dans le choix de votre futur éditeur, nous allons simplement parler de « `Emacs` » en général et tout ce qui suit sera commun à toutes les versions de `Emacs` (n'exagérons rien tout de même, les différences sont assez minimes entre les versions).

Assurez–vous simplement qu'une des deux versions de `Emacs` précédemment citées est bien installée sur votre système : vous pouvez utiliser sous linux la commande `locate emacs` (ou `locate xemacs`) ou si celle–ci fait défaut sur votre système la commande `whereis`, `which` ou `type`. Si vous ne comprenez rien à ce paragraphe, faîtes–vous aider par quelqu'un pour vérifier la présence de `Emacs` sur votre machine et/ou installez–le. On supposera à partir d'ici que tout est bien installé et fonctionnel[1].

3.2 Mode de fonctionnement

`Emacs` fonctionne à la fois en mode console et en mode graphique. Suivant l'environnement dans lequel vous vous trouvez, vous pouvez donc utiliser votre éditeur tout simplement.

En mode console, pour lancer `Emacs`, tapez simplement :

```
~$ emacs
```

[1]N'importe quelle ditribution moderne de Linux vous installera cela en deux coups de cuillère à pot. Vous pourrez même faire cohabiter toutes les versions sans difficulté pour vous donner une idée.

(le « ˜$ » est considéré comme le prompt de votre shell, vous n'avez qu'à taper `emacs` en fait)

En mode graphique (sous *X Window* généralement), ouvrez une *xterm*, où n'importe quel émulateur de terminal et tapez :

```
~$ emacs &
```

Le « & » final servant à mettre la commande en tâche de fond. `Emacs` détectera automatiquement la présence de *X11* et lancera son interface graphique. Si vous désirez ne pas l'utiliser et garder `Emacs` dans le terminal comme en mode console, il faudra le forcer :

```
~$ emacs -nw
```

où l'option `-nw` (pour « *no window* ») forcera `Emacs` à ignorer le serveur *X* et lancer `Emacs` en mode console.

En cas d'utilisation sous *X*, assurez–vous bien que votre `Emacs` ou que le terminal utilisé est bien la fenêtre active pour travailler sinon tous les caractères tapés seront ignorés :–(

Chapitre 4

`Emacs` comme éditeur de texte.

Dans tout ce qui suit, on va regarder comment fonctionne `Emacs` comme éditeur de texte. Toutefois, 80% des commandes citées vont pouvoir vous servir par la suite dans d'autres parties comme le courriel, la gestion des révisions, *etc.*, bref tout ce qui fait appel à du texte et qui est éditable (soit quasiment tout sous *Unix* :–)).

`ATTENTION` : la faute classique du débutant consiste à se mélanger les pinceaux en écrivant une commande. `Emacs` lance alors une série de « bips » sonores (ou « flashe » l'écran suivant les versions) et reste bloqué sur l'erreur. C'est très énervant lorsque l'on ne sait pas en sortir :–(

Il existe une commande miracle qui annule toute commande en cours. Si vous devez en connaître une seule *par coeur*, cela DOIT être celle–ci :

```
C-g
```

Avec cela, vous êtes paré pour attaquer la suite :–)

Dans tout ce qui suit, on adoptera la convention suivante d'écriture :

```
le raccourci clavier (la fonction LISP appelée) : son
explication.
```

4.1 Édition de texte

(*on rappelle que l'on peut utiliser la tabulation pour « finir » l'écriture d'un nom de fichier ou la commande elle–même sans raccourci.*)

`C-x C-f (C-x find-file): nom du fichier`	`ouverture d'un fichier.`
`C-x i (C-x insert-file): nom du fichier`	`insertion d'un fichier à la position courante du curseur.`
`C-x s (C-x write-file)`	`sauvegarde du tampon courant.`
`C-x C-w (C-x write-buffer): nom`	`sauvegarde dans un autre fichier.`
`C-x C-c (C-x save-buffers-kill-emacs)`	`sortie.`

S'il existe un tampon qui n'a pas été sauvegardé lors de la sortie de `Emacs`, ce dernier demandera une confirmation avant de sortir : il sera nécessaire alors de répondre explicitement « *yes* » pour confirmer la destruction du tampon.

4.2 Déplacement du curseur.

Tout `Emacs` moderne et bien configuré vous permet de vous déplacer pratiquement naturellement dans le texte à l'aide des fonctions intuitives du clavier : les flèches. Mais comme toute action sous `Emacs`, cela se traduit par l'appel d'une fonction `LISP` que, dans ce cas précis, il n'est pas forcé de connaître :–)

Toutefois, comme tout éditeur de texte digne de ce nom, `Emacs` vous en propose beaucoup plus : saut de mots, de phrases, de paragraphes, déplacement d'un bout à l'autre du tampon, *etc.* Toutes ces fonctions permettent un gain de temps extraordinaire. La connaissance de l'anglais est précieux pour se souvenir des commandes car leur traduction apporte une connaissance immédiate de leur nature, et souvent de leur raccourci. Par exemple, la commande :

```
C-f (M-x forward-char) : déplace le curseur d'un caractère à
                         droite, soit en avant.
```

Or si l'on traduit « *forward–char* », on a mot–à–mot « avant–caract » où « caract » est un diminutif de « caractères » bien sûr ("*char*" pour "*character*" en anglais). Or le raccourci clavier est `C-f`, « f » pour « *forward* » donc très facile à retenir… pour l'anglophone. On vous avait prévenu, l'informatique n'est faite que pour eux…

Par la suite, on omettra le « `C-x` » ou « `M-x` » de la commande `Emacs`.

```
C-f (forward-char)  : déplace le curseur d'un caractère à droite,
                      soit en avant.
C-b (backward-char) : déplace le curseur d'un caractère à gauche,
                      soit en arrière (« backward » en bon anglais).
C-n (next-line)     : déplace le curseur d'une ligne suivante
                      (« next »)
C-p (previous-line) : déplace le curseur sur la ligne précédente
                      (« previous »)
```

Ces quatre actions sont avantageusement remplacées par celle des flèches du clavier, bien plus simples et plus conviviales d'utilisation :–)

```
M-f (forward-word)  : déplace le curseur d'un mot vers la droite...
M-b (backward-word) : déplace le curseur d'un mot vers la gauche...
```

Un raccourci du raccourci consiste à utiliser les flèches droites et gauches avec la touche « CONTROL » enfoncée. C'est plus rapide encore :–)

Les plus curieux auront noté l'analogie évidente avec le déplacement d'un caractère. En effet, `Emacs` est logique et ne vas pas appeler un chien un chat. On garde le « f » pour « *forward* » et on change simplement sa portée : `C-`. On touche ici une petite subtilité d'`Emacs`. De façon générale, les commandes commençant par « M- » ont une portée locale (ici le caractère) et celle commençant par « C- » ont une portée plus générale, plus « supérieure » (ici le groupe de caractères puisque l'unité est le caractère, donc le mot :–)). C'est un moyen mnémotechnique de plus de se souvenir des raccourcis clavier :–)

```
C-a (beginning-of-line)     : déplace le curseur en début de ligne
                              (« begin » , début en anglais)
C-e (end-of-line)           : déplace le curseur en fin de ligne
                              (« end », fin en anglais)
```

Ceci est un exemple typique de commande réutilisable en dehors de votre éditeur, par exemple dans votre shell préféré.

```
M-a (backward-sentence)     : déplace le curseur en début de phrase
M-e (forward-sentence)      : déplace le curseur en fin de phrase
M-{ (backward-paragraph)    : déplace le curseur en début de paragraphe
M-} (forward-paragraph)     : déplace le curseur en fin de paragraphe

C-v (scroll-up)             : avance d'un écran de texte
M-v (scroll-down)           : recule d'un écran de texte.
```

Ces deux commandes sont avantageusement remplaçables par les touches « *page up* » et « *page down* », c'est–à–dire les touches à droite de la touche « *home* » ou « début » suivant les claviers et la touche « fin » (ou « *end* »). C'est le mini–pavé au–dessus des flèches de direction.

```
M-> (end-of-buffer)        : déplace le curseur en fin de tampon.
M-< (beginning-of-buffer)  : déplace le curseur en début de tampon.
```

On va voir ici la première différence entre GNU Emacs et XEmacs. Il existe des autres raccourcis, suivant la version de Emacs que vous utilisez : Pour se déplacer en début et en fin de tampon, il suffit de taper sur la touche « début » (« *home* ») et « fin » (« *end* ») sous GNU Emacs tandis qu'il faut assortir cette touche de la touche « CONTROL » sous XEmacs.

```
(rien) (goto-line) numéro  : place le curseur à la ligne <numéro>
```

En fait, il existe un raccourci pour XEmacs (GNU Emacs ne l'implémente pas...). Il s'agit de M-g.

```
C-l (recenter)            : place la ligne actuelle au centre de
                            l'écran.
```

Vous allez me dire, si vous avez tenu jusque là, que c'est bien beau, mais cela fait beaucoup à retenir, que l'on se mélange les doigts, et que (argument suprême !), on a une souris et des ascenseurs et que l'on va aussi vite ainsi...

Non. Non. Non. Pour ce qui est de la maîtrise de ses doigts, seul l'entraînement pourra vous dépanner... *Cent fois sur l'ouvrage...* Quant à l'utilisation de la souris et des artifices de *X*, vous verrez qu'à l'usage, une fois maîtrisé l'ensemble des raccourcis, que vous serez bien plus rapide avec le clavier seul. Vous verrez...

4.3 Suppression de texte

```
DEL  (backward-delete-char) : supprime le caractère précédent.
C-d  (delete-char)          : supprime le caractère à la position
                              du curseur
```

Il va de soit qu'un Emacs bien configuré permet ces opérations à l'aide des touches « *backspace* » (la touche fléchée au dessus de la touche « Entrée »). La touche « *DEL* » est la touche « Suppr » des clavier *azerty*.

```
M-DEL (backward-kill-word) : supprime le mot précédent.
M-d   (kill-word)          : supprime le mot sous le curseur
C-k   (kill-line)          : supprime la fin de la phrase à partir
                             du curseur.
M-k   (kill-sentence)      : supprime la phrase sous le curseur.
C-w   (kill-region)        : supprime une région (voir § suivant)
```

4.4 Marquage et région

Un éditeur de texte n'en serait pas un si l'on ne pouvait travailler exclusivement sur une région du texte sur laquelle on travaille.

```
C-ESPACE (set-mark-command) : marque le début d'une région.
```

Pour prolonger la région, il suffit à partir de la commande précédente, de parcourir le texte à l'aide de n'importe quelles commandes de déplacement de Emacs précédemment citées, en amont ou en aval. La région s'arrête alors sous le curseur et englobe l'ensemble des lignes et caractères comptés à partir du début de la région.

Par exemple, aller dans un paragraphe, se placer en début d'un mot, marquer la région, se déplacer de trois mots et taper C–w effacera les trois mots. Avec un peu de pratique, cette opération requière un temps très court. . .

Attention, sous XEmacs, par défaut, la région passe en grisé, ce qui permet un contrôle visuel. Sous GNU Emacs, par défaut, la coloration ne change pas. Il faut donc faire attention à ce que l'on fait où activer l'option par défaut (voir § *configuration avancée*).

```
C-x h (mark-whole-buffer)  : marque tout le tampon
M-h   (mark-paragraph)     : marque le paragraphe
M-g   (fill-region)        : reformate le paragraphe.
```

Cette dernière commande est particulièrement utile : par défaut, comme tous les éditeurs *Unix*, Emacs travaille par ligne et n'introduit pas de retour chariot lorsqu'il arrive au bord de l'écran droit. Si l'on allonge la largeur de la fenêtre, le texte se reformate automatiquement, comme pour un butineur HTML. Cela n'est généralement pas gênant, sauf pour le courriel, les niouzes ou certains textes que l'on veut spécialement formater. Emacs découpe alors automatiquement le texte pour qu'il tienne dans les quatre–vingt colonnes réglementaires :–)

```
M-w (kill-ring-save)       : copie une région sans destruction.
```

L'avantage de cette dernière commande sur `C-w` est que la région copié reste en place.

```
C-y (yank)                 : recopie sous le curseur la région
                             copiée.
```

4.5 Copier, coller et morcellement

Pour copier, il faut marquer une région comme précédemment dit.

```
c-ESPACE (set-mark-command)  : marque le début d'une région
C-y      (yank)              : colle la région sous le curseur

C-x r k  (kill-rectangle)    : supprime un rectangle en l'enregistrant
C-x r d  (delete-rectangle)  : supprime un rectangle sans l'enregistrer
C-x r y  (yank-rectangle)    : insère le dernier rectangle enregistré.
C-x r o  (open-rectangle)    : insère un rectangle de blancs.
```

Ces quatre opérations sont très pratiques pour des couper dans un texte comportant des données sous forme de tableaux. Une faiblesse de `Emacs` est qu'il n'est pas prévu de copier/coller de rectangle. Pour ce faire, il est nécessaire de couper la partie à coller, la recopier immédiatement puis aller la recopier ailleurs. Ce n'est pas pratique... On doit bien perdre une demi–seconde pour les moins rapides :–)

Comment est défini un rectangle ? Tout simplement de la même manière que la région (Ouf !...). Vous délimitez le début de votre rectangle par son coin supérieur gauche par `C-ESPACE` puis allez vous positionnez sur son coin inférieur droit. Il ne reste alors plus qu'à taper une des commandes précédentes qui n'agira que sur la partie rectangle. Les personnes utilisant un marquage de couleurs pour les régions ne doivent pas se laisser surprendre par le fait que tous les caractères situés entre le début du rectangle et la fin seront marqués, y compris ceux n'appartenant pas au rectangle proprement dit. Le marquage d'une région marque l'ensemble des caractères de cette dernière : seule la commande suivante pourra trier l'information, c'est–à–dire traiter toute la région (`C-w` par exemple) ou une partie rectangle (avec `C-x r k` par exemple). `Emacs` n'est pas omniscient, malgré toutes ses qualités !

Il arrive parfois que l'on désire insérer un jeux de caractères identiques devant un certain nombre de lignes. L'exemple caractéristique est de vouloir commenter un certain nombre de ligne dans un programme ou fichier de configuration. Le commentaire est typiquement le caractère « # » (`C`, `C++`, *Perl*, Shell, ...) , « % » (*TeX*, *LaTeX*, ...) et « ; ; » (`LISP`). Pour cela, il faut taper :

```
C-x r t
```

à la fin de la région. Il apparaît alors dans la ligne de commande le prompt dans lequel on tape le(s) caractère(s) voulu(s) puis la touche « Entrée ».

4.6 Annulation

Tout un paragraphe sur l'annulation, quelle richesse ! Emacs mémorise tout et donc vous autorise pas mal de retours en arrière. En fait, tant que vous n'avez pas sauvegardé votre tampon, vous pouvez annuler TOUTES les opérations que vous avez précédemment effectuées. Pas mal, non ?

```
C-g   (keyboard-quit)     : met fin à la commande en cours
                             (PAR COEUR!!!)
C-x u (advertised-undo)   : annule la dernière action.
                             Peut être répétée.
```

En fait, cette combinaison de touche est malaisée, si tant est que l'on désire plus d'une annulation (essayez et vous verrez...). Elle est alors avantageusement remplacée par C _ (la touche « _ », dite « *underscore* » en bon franglais). Il suffit de la laisser appuyer : ATTENTION, c'est rapide...

Enfin, pour revenir directement dans l'état initial du tampon, il faut taper :

```
C-x revert-buffer
```

4.7 transpositions et casse

```
C-t     (transpose-char)       : transpose deux caractères
M-t     (transpose-word)       : transpose deux mots
C-x C-t (transpose-lines)      : transpose deux lignes
 (rien)  (transpose-sentences) : transpose deux phrases
 (rien)  (transpose-paragraphs): transpose deux paragraphes

M-c     (capitalize-word)      : met la lettre sous le curseur
                                 en majuscule sinon la première
                                 lettre du mot suivant.
M-u     (upcase-word)          : met le prochain mot en majuscule.
M-l     (downcase-word)        : met le prochain mot en minuscule.
C-x C-u (upcase-region)        : met la région en majuscule.
C-x C-l (downcase-region)      : met la région en minuscule.
```

4.8 Recherche incrémentale

Un éditeur de texte sans un moteur de recherche d'expression évoluée n'est pas un réel éditeur de texte. On distingue deux types de recherche : la recherche d'une chaîne définie et d'une expression rationnelle.

4.8.1 Chaînes simples

On appelle « chaîne simple », toute expression dont on connaît chaque caractère, ou au moins le début de la chaîne de caractères. Pour rechercher dans ce cas–là, il suffit de taper :

```
C-s    (isearch-forward)   : lancement ou répétition d'une recherche
                             incrémentale vers l'avant.
C-f    (isearch-backward)  : lancement ou répétition d'une recherche
                             incrémentale vers l'arrière.
Entrée                     : Quitte une recherche réussi.
C-g    (keyboard-quit)     : quitte une recherche et revient au point
                             de  départ.
Suppr                      : supprime un caractère incorrect de la
                             chaîne recherchée.
```

Concrètement, pour rechercher une chaîne de caractères, vous taper `C-s` (ou `C-f` suivant le sens de la recherche) puis le début de la chaîne. Si vous cherchez les occurrences du terme « Linux », vous avez deux solutions :

– soit vous tapez « Lin » et auquel cas, la recherche sera sensible automatiquement à la casse car vous avez mis un jeu de casse différent dans la même chaîne.
– soit vous tapez « lin » et la recherche sera indépendante de la casse. Toutes les occurrences contenant les trois lettres seront trouvées, indépendamment de leur casse.

Pourquoi ne pas taper toute la chaîne ? C'est une question de méthode. Si vous recherchez la chaîne « Linux » dans un texte, la probabilité de trouver un terme contenant « lin » est faible (mais loin d'être nulle). Or vous vous apercevrez que dès que vous tapez le début de votre chaîne à rechercher, `Emacs` commence déjà la recherche et se place automatiquement sur le mot (ou la partie de mot) correspondant. Souvent, il est plus aisé (et moins fatigant !) d'itérer la recherche suivante (en appuyant une nouvelle fois sur `C-s` ou `C-f` suivant ce que vous avez tapé au début) que de finir la chaîne de caractères. On répète que l'informatique est là pour faire le boulot à votre place, alors profitez–en ! On se balade ainsi dans tout le tampon jusqu'à avoir trouvé la bonne chaîne.

4.8.2 Expressions rationnelles

On ne connaît pas toujours la description exacte de la chaîne recherchée. On a vu précédemment que l'on pouvait déjà rechercher une chaîne qui comportait des majuscules ou des minuscules. C'est un début de recherche « dans le doute » mais cela reste très limitée.

Les expressions rationnelles (« *regular expression* » en anglais[1]) sont un des « dadas » d'*Unix* et d'une myriade d'outils tournant autour (`sed`, `awk`, *Perl*, `vi`, *etc...*). Cela permet de désigner une expression par ce qu'elle pourrait contenir, bref son gabarit et non plus sa forme exacte. `Emacs` est alors capable de rechercher toutes les expressions se rapportant à ce gabarit. La seule difficulté consiste à apprendre la syntaxe du gabarit. Après, c'est enfantin... et d'une puissance redoutable.

```
M-C-s RETURN (re-search-forward)    : recherche en avant à l'aide d'une
                                      expression rationnelle.
M-C-f RETURN (re-search-backward)   : recherche en arrière à l'aide d'une
                                      expression rationnelle.
M-C-s  (isearch-forward-regexp)     :  recherche incrémentale en avant à l'aide
```

[1] La traduction en *expression régulière* est évidemment une faute...

```
                                           d'une expression rationnelle.
M-C-f  (isearch-backward-regexp)   :  recherche incrémentale en
                                           arrière à l'aide d'une
                                           expression rationnelle.
```

Pour la désignation du gabarit, voici la syntaxe :

```
^     : début de ligne
$     : fin de ligne
.     : n'importe quel caractère
.*    : n'importe quelle suite de caractères, même vide.
\<    : début de mot
\>    : fin de mot
[]    : n'importe quel caractère contenu dans l'intervalle.
```

Cela parait compliqué mais cela est assez simple. Il suffit de « penser » sa recherche. Par exemple, je veux tous les mots ayant les propriétés suivantes : en tête de ligne, commençant soit par une lettre minuscule et comportant 5 caractères. Il suffit dès–lors de décomposer sa pensée :

```
- en tête de ligne                                  --->  ^
- commençant (donc début de mot)                    --->  \<
- par une minuscule donc intervalle entre a et z   --->  [a-z]
- comportant 5 caractères (on a déjà le premier ) --->  ....
```

soit l'expression rationnelle : `^\<[a-z]....`

Avouez qu'il est plus facile de la trouver que de la lire, n'est–ce pas ? Le système est extrêmement puissant et facile d'utilisation. Essayez–le et vous l'adopterez.

Une dernière petite chose : pour chercher les caractères \, <, >, ., ^, [et], il faut les « protéger » de façon à ce que Emacs ne les interprète pas comme une commande d'expression rationnelle. Pour ce faire, il faut utiliser le \ (« *anti–slash* » ou barre oblique) accolé au caractère que l'on veut protéger. Il faut répéter l'opération autant de fois que de caractères spéciaux sont en jeu et doivent être reconnu comme des caractères normaux.

4.9 correction orthographique

Une des nombreuses fonctionnalités de Emacs est d'offrir une interface à un correcteur orthographique bien connu du monde *Unix*, à savoir ispell. Alors, pourquoi s'en priver, d'autant plus qu'il est si facile de s'emmêler les doigts sur un clavier :–)

ispell offre un choix varié de dictionnaire, dont un français (deux même !). Par défaut, le dictionnaire est évidemment... anglais. C'est bon, je vois que certains commencent à suivre :–). Il est aisé d'en changer et d'installer un dictionnaire plus en accord avec la langue de Molière. On peut même en cumuler plusieurs, suivant les besoins. Leurs installations concernent votre distribution. Je ne saurai trop vous recommander Debian car toutes ces opérations sont gérées très simplement, et sans effort.

On supposera ici que votre dictionnaire est bien installé.

```
M-$        (ispell-word)      : vérifie le mot situé sous ou après
                                le curseur.
(rien)     (ispell-region)    : vérifie tous les mots d'une région.
(rien)     (ispell-buffer)    : vérifie tout le tampon.
(rien)     (ispell-message)   : ne vérifie que le message dans le cas
                                d'un courriel.
C-u M-$    (ispell-continue): reprend le processus ispell précédemment
                                arrêté par un C-g
```

Lors d'une vérification, `ispell` s'arrête sur tous les mots qu'il ne connaît pas et prend au choix une des deux attitudes suivantes. Soit le mot est correct, mais n'appartient pas à son dictionnaire (comme un nom propre, un sigle, *etc.*) et vous lui demandez de passer au suivant avec la touche « espace », soit le mot est erroné et il vous en propose plusieurs corrections. Chaque mot est précédé d'un chiffre entre parenthèse et il vous suffit de taper le chiffre correspondant à la bonne correction et `Emacs` fera la substitution. Vous n'avez plus d'excuse pour laisser des fautes de frappes après cela...

Chapitre 5

Éditions multiples et tampon

5.1 Gestion d'un tampon multiple

Emacs peut gérer plusieurs tampons en même temps. En fait, comme pour la longueur d'un tampon[1], la limite est physique : tant que vous avez assez de mémoire, vous pouvez ouvrir un nouveau tampon.

Voilà qui est bien utile pour travailler : il est souvent nécessaire de travailler simultanément sur plusieurs fichiers en même temps. Sous Emacs, pas de problème. Cela permet même de laisser de côté un fichier ouvert pour y revenir plus tard, d'ouvrir un brouillon, bref une liberté totale d'action.

Comment ouvrir simultanément plusieurs tampons ? en fait, il suffit d'ouvrir un nouveau fichier et ce dernier prend automatiquement la place du tampon courant. l'ancien est conservé et n'apparaît plus à l'écran.

```
C-x C-f (find-file)        : ouvre un nouveau fichier
C-x b   (switch-to-buffer) : met à l'écran le tampon spécifié
```

Cette dernière commande permet de passer d'un tampon à l'autre. Par défaut, Emacs propose le dernier tampon ouvert : il suffit alors de taper sur « Entrée ». Sinon, taper le début du nom du tampon et « tabulation » pour avoir la fin. . . . puis « Entrée ».

```
C-x C-b (list-buffer)     : fait apparaître la liste des tampons
                            en mémoire.
```

Il suffit de cliquer avec la souris (bouton du milieu) pour faire apparaître le tampon désiré. On peut aussi rendre active la fenêtre de la liste (M-v) puis déplacer le curseur sur le bon tampon et valider.

```
C-x k  (kill-buffer)     : détruit un tampon.
```

Par défaut, Emacs propose de détruire le tampon en édition. Si l'on appuie sur la touche « Entrée », c'est ce qu'il se passera. Sinon, écrire le nom d'un autre tampon et valider.

[1]Ceci n'est pas tout à fait exact en fait : la taille d'un tampon est limitée à un peu plus de 100 méga–octets. C'est très suffisant pour une utilisation normale.

Le fait de jouer avec plusieurs tampons implique de la part de `Emacs` une gestion indépendante des tampons. Dès–lors, vous ne devez pas quitter `Emacs` n'importe comment car il se peut qu'un tampon n'ait pas été sauvegardé. En cas d'accident (sortie du serveur *X* sans sortir d'`Emacs`, « *kill* » malheureux, « *crash* » du système, *etc...*), il suffit souvent de relancer `Emacs`. Lors que vous réessayerez d'ouvrir le fichier accidenté, `Emacs` vous signalera qu'il existe une version mal enregistrée et vous proposera de la garder. Il suffira de taper :

```
M-x recover-file (le nom du fichier incriminé)
```

et tout sera réparé.

5.2 Subdivision de l'espace de travail

Il est parfois pratique de visualiser simultanément deux textes en parallèle. Cela permet un contrôle visuel des différences par exemple (quoiqu'il existe des outils bien plus puissants pour effectuer cette opération : *cf.* `diff` et `man diff` pour plus d'informations). Or ouvrir deux tampons, c'est bien joli mais encore faut–il pouvoir les visualiser simultanément.

`Emacs` peut le faire facilement. Mieux encore, il permet de visualiser plusieurs tampons simultanément. La limite, là encore, est matériel : c'est l'écran... essayer de visualiser plus de quatre tampons devient « clownesque ». Quant à l'utilité... Enfin, essayez et vous verrez :–)

```
C-x 2   (split-window-vertically)   : découpe le tampon en deux fenêtres
                                      superposées.
C-x 3   (split-window-horizontally): découpe le tampon en deux fenêtres
                                      juxtaposées.
C-x o   (other-window)              : passe à la fenêtre suivante.
C-x 0   (delete-windows)            : détruit la fenêtre courante.
C-x 1   (delete-other-windows)      : détruit toutes les fenêtres sauf
                                      la fenêtre courante.

C-x ^   (enlarge-window)            : augmente la hauteur
(rien)  (shrink-window)             : diminue la hauteur
```

Ces deux dernières commandes sont tout bêtement remplacées par l'utilisation de la souris sous *X*. Il suffit de cliquer sur la barre d'état avec le bouton droit et d'agrandir ou diminuer suivant les besoins. C'est pour le coup beaucoup plus pratique et rapide :–)

Rien n'empêche de cumuler ensuite des fenêtres juxtaposées avec des fenêtres superposées, si ce n'est la lisibilité. En tout cas, amusez–vous bien...

5.3 Le plus de *X Window*

X Window (que l'on appelle aussi *X11*) est l'environnement graphique standard d'*Unix*. On a alors accès aux fenêtres avec ascenseur, la gestion de la souris, *etc.* même si la plupart de ces avantages sont disponibles en mode console sous Linux.

Emacs étant né un peu avant *X11*, ou au moins à une période où tout le monde n'avait pas forcément les moyens d'avoir accès à un serveur *X*, tous ces avantages ont été implémantés sous les formes vues précédemment : tampons multiples, déplacements aisés au sein d'un tampon et des autres, gestion de la souris, *etc...* donc en fait utiliser Emacs sous *X* n'apporte pas ou peu d'avantages supplémentaires. On peut simplement signaler l'utilisation des cadres (« *frame* » en bon anglais).

Le cadre est une nouvelle fenêtre graphique qui vient se superposer à votre ancien tampon. Elle a alors toutes les propriétés d'une fenêtre sous *X* et peut se déplacer à la souris, se redimensionner (j'y vois là d'ailleurs le seul avantage de *X*, avec de belles couleurs !), *etc.* Bref, un objet graphique comme un autre sous votre gestionnaire de fenêtres. Ainsi, on peut garder sous la souris tous les tampons que l'on désire... On peut les icônifier aussi. C'est pratique mais n'apporte pas de réelles et nouvelles fonctionnalités à votre Emacs.

```
C-x 5 2  (make-frame)    : crée un nouveau cadre.
C-x 5 o  (other-frame)   : active le cadre suivant
C-x 5 0  (delete-frame)  : détruit le cadre actif.
```

5.4 Documentation

Emacs a un avantage énorme sur beaucoup de logiciels : il intègre une masse faramineuse de documentation. Cette documentation est d'ailleurs si riche qu'elle perturbe souvent le débutant : c'est la raison principale pour laquelle j'ai écrit cette documentation. Une documentation un peu simpliste permet de mettre plus facilement le pied à l'étrier avant de monter réellement en selle.

La documentation de Emacs est au format « info ». Elle autorise donc une gestion de liens hypertextes, comme le HTML. Il suffit donc soit de cliquer soit de le faire avec le clavier (M-v puis « Entrée »). C'est donc très simple de chercher une information.

Il existe quand même un écueil de taille pour nombre d'anglophobes : la documentation est exclusivement en anglais... Mais ne vous inquiétez pas : si vous possédez des rudiments, et à l'aide des mots–clés que je me suis efforcé de traduire tout au long de cette documentation, vous devriez vous en sortir...

J'ai mis un drapeau *[initié]* lorsque la documentation ne concerne pas du tout le débutant et *[tout public]* sinon. Attention, ce dernier drapeau ne veut pas dire pour autant que le sujet est facile mais qu'il contient des informations essentielles pour débuter.

C-h t : lance un tutoriel. Normalement, le présent document devrait déjà vous avoir sérieusement dégrossi. *[tout public]*

C-h p : liste les différents thèmes traités par Emacs puis la liste des paquetages correspondants. *[initié]*

C-h F : lance la « Foire aux questions », dites FAQ. Ce sont les questions les plus fréquemment posés par les débutants sur les différents groupes de discussions (« *Frequently Asked Questions* »). Lecture plus qu'indispensable[2]. *[tout public]*

[2] Toute question posée dans un groupe de discussions (quel qu'il soit d'ailleurs) ET contenue dans la FAQ du groupe est une cause fréquente d'échanges houleux (pour rester poli...). Dîtes–vous bien une chose : voir passer pour la centième fois (des fois, bien plus) la même question (même légitime) est doublement énervant : cela bloque la bande–passante du groupe en augmentant son trafic pour des raisons stupides et cela met en évidence que le travail de tout le groupe des personnes qui

`C-h i` : lance le manuel info. Ce n'est pas spécifique à `Emacs` mais vous en apprendrez sur tout votre système. *[tout public]*

`C-h C-f` : information au sujet d'une commande *[tout public]*

`C-h C-k` : information au sujet de la commande associée à une séquence de touches. *[tout public]*

`C-h C-c` : Licence de `Emacs` *[A LIRE]*

`C-h C-p` : Information sur le projet GNU

`C-h C-w` : La non–garantie de `Emacs` *[A LIRE ABSOLUMENT]*

Il existe d'autre part un excellent ouvrage en français sur `GNU Emacs`, même s'il a un peu vieilli, surtout pour les à–côtés (courriel notamment).

« *Introduction à `GNU Emacs`* », Debra Cameron & Bill Rosenblatt. traduction de Jean–Baptiste Yunès. Édition *O'Reilly* (2ième édition) ISBN : `2-84177-015-X`

réalisent la FAQ n'est pas pris en compte. C'est un travail souvent pénible, peu glorieux et terriblement prenant. Alors, par pitié et par respect pour ce travail ingrat, lisez la FAQ : il y a de toute façon neuf chances sur dix pour que la solution à votre problème y soit…

Chapitre 6

Conclusion

Voilà, on arrive à la fin. J'espère que cette documentation vous aura permis de lever un peu le voile sur le « mystère `Emacs` ». J'ai volontairement réduit la présente documentation aux aspects élémentaires de `Emacs`, sans entrer dans toutes les fonctions. Peut–être ai–je trop réduit, je n'en sais rien mais normalement, muni de ce bagage, vous devriez être capable de poursuivre votre initiation en solitaire, armé de votre seul `Emacs` :–)

Si la demande existe, je veux bien élargir cette documentation à des aspects plus techniques de `Emacs`, à savoir la configuration du `.emacs` et du `.Xdefaults`, le courrier électronique, les groupes de discussions, le gestionnaire de révision, la programmation, *etc.* bref, tout ce qui fait que `Emacs` est un « environnement de textes » à part et qu'il déclenche tant de passions. . .