Emacs il le fait !

Benjamin Drieu (*bdrieu@april.org*) 16 octobre 1999

Comment lire mes mails et en envoyer ? Comment lire les news ? Comment manipuler les répertoires à la Midnight Commander ? Comment compiler sans basculer dans un terminal ? Est-ce possible de compiler un document LaTeX sans douleur ? Comment puis-je accéder à un site FTP ? Comment naviguer sur le web ? Comment on utilise CVS ? Existe-t-il un agenda ? Comment imprimer un document texte ? Comment vérifier l'orthographe de mon texte ? Où trouver un interpréteur Lisp ? Il y a des jeux ? Comment on lance VI ? Une seule réponse à toutes ces questions ainsi qu'à bien d'autres : "Emacs le fait !"

Contents

1 Introduction

1.1 A propos de cette documentation

Cette documentation n'a pas pour but d'être exhaustive, ni précise. Je l'ai conçue comme étant une ouverture sur la face cachée d'Emacs. Il manque cependant une partie essentielle, qui explique comment se documenter, et trouver de l'information sur la configuration (notamment, comment remplir et organiser son fichier `.emacs`). De plus, la plupart des sections (si ce n'est toutes) manquent cruellement d'exemples pratiques, et se limitent parfois à une simple énumération de commandes.

Pour une utilisation *normale* d'Emacs (en tant que traitement de textes), je vous conseille de vous reporter à une première documentation, disponible sur le site web d'APRIL (`http://www.april.org/groupes/doc/`). Pour une utilisation avancée d'Emacs (programmation de macros en Lisp), reportez vous à l'introduction à la programmation *Emacs-Lisp* disponible sur le site ftp de Gnu ou un de ses miroirs (`ftp://ftp.gnu.org/pub/gnu/emacs-lisp-intro-1.05.tar.gz`).

1.2 Qu'est-ce qu'Emacs ?

Emacs est un éditeur de textes interactif plein écran. Emacs signifie *Editor MACroS*, et a été développé par Richard Stallman à partir de 1976 au MIT sur un projet de Guy Steele. A l'origine, Emacs était un ensemble de macros pour l'éditeur TECO (*Text Editor and COrrector*) sur un PDP-10.

Une différence majeure par rapport à d'autres éditeurs est qu'Emacs est en fait un interpréteur *Lisp*. La plupart des commandes sont des commandes Lisp par dessus un noyau C qui gère l'affichage et l'interprétation des commandes Lisp.

Emacs est de plus auto-documenté, ce qui facilite l'apprentissage et permet à l'utilisateur de l'apprendre sans imprimer de manuel.

1.3 Conventions utilisées

- `RET`: frappe de la touche `Enter`.
- `DEL`: frappe de la touche `Delete`.
- `SPC`: frappe de la touche `Espace`.
- `ESC`: frappe de la touche `Esc`.
- `C-x`: frappe de la touche `Ctrl` et de la touche `x` simultanément (la casse n'est pas importante).
- `M-x`: frappe différée de la touche `Esc` puis de la touche x, ou sur certains claviers de la touche `Alt` et de la touche `x` simultanément
- `M-x toto`: même chose que précédemment, puis saisie de la séquence de caractères `toto`.
- *chaîne*: une chaîne de caractères
- *regexp*: une expression régulière (en anglais, *regular expression*)

2 Internet

2.1 Comment lire ses mails et en envoyer ?

Je présenterais ici RMail, mon lecteur de mails préféré. Mais il en existe au moins deux autres tout aussi puissants si ce n'est plus : MH et VM. Si quelqu'un est intéressé par l'écriture d'une courte documentation sur l'un ou l'autre je suis preneur.

2.1.1 Introduction

RMail possède toutes les fonctionnalités d'un lecteur de mails classique comme *pine*, *elm* ou même (horreur) *netscape*. Mais il offre toutes les facilités d'Emacs. On invoque RMail par `M-x rmail`, ce qui fait apparaître un buffer nommé `RMAIL`.

2.1.2 Mails

Les mails sont organisés en fichiers au format RMail, dont l'utilisation est assez semblable aux *folders* d'autres lecteurs de mails. Il existe des commandes pour déplacer les mails d'un fichier à un autre.

Chaque mail contenu dans un de ces fichiers peut posséder un attribut nommé *label* :

- **unseen** : ce mail n'a pas encore été lu
- **deleted** : ce mail a été marqué en vue d'un effacement
- **filed** : ce mail a été copié dans un autre folder ou un fichier
- **answered** : l'utilisateur a répondu à ce mail
- **forwarded** : l'utilisateur a envoyé ce mail à une autre personne
- **edited** : ce mail a été édité
- **resend** : ce mail a déjà été envoyé plusieurs fois à cause d'erreurs
- un label spécifié par l'utilisateur afin de classer ses mails

Le déplacement dans un message se fait grâce aux commandes standards d'Emacs dans le buffer `RMAIL`, ou à l'aide des commandes suivantes :

- `SPC`: descend d'une page (comme `C-v`)
- `DEL`: monte d'une page (comme `M-v`)
- `.`: va au début du message (comme `M-<`)

2.1.3 Listes de mails

Les mails peuvent être visualisés dans une liste (buffer `RMAIL-summary`). Cette liste permet de voir la date d'un mail, ainsi que son expéditeur et son sujet. Le déplacement dans cette liste permet d'activer chacun des mails existants, et de lui appliquer des commandes :

- `d`: marque le message comme à détruire
- `u`: enlève la marque de destruction du message
- `x`: détruit les messages marqués (purge)
- `n`: lit le message suivant
- `p`: lit le message précédent
- `>`: lit le dernier message
- `<`: lit le premier message
- `M-s` *regexp*: cherche parmi les messages suivants celui contenant l'expression régulière *regexp*

2.1.4 Commandes d'envoi de mail

- `m`: envoi d'un nouveau mail, saisi dans un buffer en mode mail (`*mail*`)
- `r`: réponse au mail sélectionné
- `f`: *forward* d'un mail sélectionné
- `M-m`: tente d'expédier à nouveau un mail dont l'expédition n'a pas réussi

2.1.5 Mode mail

A faire...

2.1.6 Manipulations sur fichiers RMail

RMail conserve ses mails dans des fichiers au format *RMail*. Emacs permet des manipulations sur ces fichiers, comme l'import de fichiers RMail ou l'export vers un autre fichier.

- `i` *fichier*: lance RMail sur ce fichier. Ceci permet de consulter des folders différents
- `o` *fichier*: exporte le fichier au format *RMail*
- `C-o` *fichier*: exporte le fichier au format mail natif du système
- `w`: extrait le corps du message dans un fichier dont le nom est le sujet

2.1.7 Variables de RMail

Ces variables sont à fixer dans le fichier d'initialisation d'Emacs (`.emacs`) pour être prises en compte à chaque démarrage.

- `rmail-delete-after-output`: si cette variable est fixée à `t`, alors tout mail exporté dans un fichier est marqué automatiquement comme à effacer
- `rmail-file-name`: cette variable est le nom du fichier qui contient les mails reçus lors de la consultation de la boîte aux lettres
- `rmail-primary-inbox-list`: cette variable permet d'utiliser plusieurs fichiers comme boîtes aux lettres. Par exemple :

  ```
  ("/var/mail/drieu" "/var/mail/http")
  ```

- `rmail-output-file-alist`: cette variable permet de fixer les fichiers dans lesquels un mail sera exporté par défaut. C'est une liste d'éléments (*a-list*) au format: `"(`*regexp* `.` *nom*`)"`. Par exemple:

  ```
  (("^From: celine@" . "~/Mail/Celine")                     ; Mails de Céline
   ("^\\(From\\|To\\): .*@risc-tech\\.com" . "~/Mail/Risc")  ; Mails professionnels
   ("^Subject: \\[GTK\\]" . "~/Mail/Gtk"))                  ; Mailing-list GTK
  ```

- `rmail-preserve-inbox`: si cette variable est fixée à `nil`, alors les messages contenus dans le fichier lu par RMail au démarrage (généralement `/var/mail/`*login*) sont effacés à chaque fois que RMail effectue une lecture dessus.

2.1.8 RMail et Pop3

Emacs peut encore faire plus fort : il connaît le protocole Pop3 par le biais de *movemail*, et permet ainsi à l'utilisateur d'accéder à son courrier par un serveur de mails. L'utilisation de Pop3 se fait de manière transparente. Pour que cela fonctionne correctement, il est nécessaire de passer par quatre étapes :

1. Le premier pas est de compiler Emacs avec l'option *movemail*. Cela se fait au moment de la configuration du logiciel, en passant l'argument `-with-pop3` au script `configure`. Par exemple :

```
./configure --with-pop3
```

2. Emacs fera ensuite appel au programme *movemail* pour rapatrier les mails à partir du serveur Pop3. Il est donc nécessaire de l'installer (c'est un programme relativement standard, il devrait déja être installé sur votre station).

3. Ensuite, il est nécessaire de spécifier quel est le serveur de mails utilisé. Affecter une valeur à la variable d'environnement `MAILHOST` suffit à satisfaire ce besoin. Par exemple :

```
export MAILHOST=mailhost.cs.univ-paris8.fr
```

4. Ensuite, il suffit de préciser à Emacs où télécharger le mail. La variable `rmail-primary-inbox-list` le permet. C'est elle qui spécifie à Emacs où trouver les fichiers contenant les mails (boîtes aux lettres).

   La syntaxe d'une boite aux lettres accessible par Pop3 est spéciale. Elle est : `"po:`*utilisateur*`"`. Lorsque RMail essaiera de lire ce fichier, il s'authentifiera sur le serveur spécifié par la variable `MAILHOST`, avec comme nom d'utilisateur *utilisateur*.

   Le serveur de mail demandera alors un mot de passe, qui est renseigné par la variable `rmail-pop-password`. Si vous l'insérez dans votre fichier d'initialisation et que vous partagez votre poste de travail avec d'autres utilisateurs, n'oubliez pas de prendre les mesures de sécurité nécessaires.

   Par exemple, si je veux à la fois lire mon mail local, et utiliser mon compte *benj* sur le serveur de mail, j'insère dans mon fichier `.emacs` :

```
(set-variable 'rmail-primary-inbox-list '("var/mail/drieu"
                                           "po:benj"))
(set-variable 'rmail-pop-password "rêvetoujoursmonami")
```

2.1.9 BBDB

BBDB ("Big Brother Database") est une base de données *rolodex* intégrée aux lecteurs de mails et de news d'Emacs. Son but est de permettre de tenir à jour une liste de personnes (carnet d'adresses).

Des informations peuvent être trouvées sur BBDB à l'URL `http://www.jwz.org/bbdb/` , notamment les variables permettant de personnaliser le mode BBDB afin de répondre aux besoins de l'utilisateur.

La base de données de BBDB est composée d'un *enregistrement* (ensemble de variables) par personne. Chaque enregistrement contient par défaut plusieurs champs:

- name : le nom complet de la personne
- AKA : surnom de la personne
- company : société ou raison sociale

- net : adresse électronique
- adress : adresse physique
- phone : numéro de téléphone
- notes : notes sur la personne

La base de données est enregistrée dans le fichier `~/.bbdb` (ce nom peut être modifié grâce à la variable bbdb-file), et peut être éditée manuellement (il s'agit de code Lisp).

Mode BBDB Le mode BBDB permet d'éditer la base de données. Il permet d'apporter des modifications aux champs de la base par le biais du buffer `*BBDB*`, qui est normalement créé grâce à la fonction bbdb (appel par `M-x bbdb`).

Cette fonction demande une expression régulière qui permet de filtrer la consultation des enregistrements.

Les commandes du mode BBDB sont :

- `n` : déplace le point sur l'enregistrement suivant
- `p` : déplace le point sur l'enregistrement précédent
- `e` : édite le champ courant (situé sous le point)
- `;` : édite les notes d'un enregistrement
- `d` : retire l'enregistrement de la base (notons que cela n'est pas définitif tant que la base n'a pas été enregistrée)
- `t` : bascule entre un affichage mono-ligne et un affichage multi-ligne de l'enregistrement. Dans le cas où cette fonction est appelée avec l'argument `t` (par exemple dans le fichier d'initialisation), tous les enregistrements sont concernés
- `s` : enregistre la base de données dans le fichier spécifié par la variable bbdb-file
- `i` : affiche le fichier info du mode BBDB (s'il a été installé)
- `q` : fait disparaître le buffer `*BBDB*` sans pour autant le détruire

Utilisation de BBDB dans un lecteur BBDB est interfaçable avec les lecteurs de mails les plus courants d'Emacs (RMail, VM, MH, Gnus). Son utilisation est très simple, et se fait par le biais de deux raccourcis communs à tous ces lecteurs de mails :

- `:` : cette fonction permet de visualiser l'enregistrement correspondant à l'expéditeur du mail sélectionné. Si aucun enregistrement ne correspond à l'expéditeur, alors BBDB demande la création d'un enregistrement
- `;` : cette fonction permet de modifier les notes d'un enregistrement. Si aucun enregistrement ne correspond à l'expéditeur du mail, alors BBDB demande la création d'un enregistrement

BBDB est également capable de créer automatiquement des enregistrements à chaque fois qu'il reçoit un mail d'une personne inconnue (ce que je déconseille personnellement).

Utiliser BBDB pour envoyer des mails L'interface d'envoi de mails sous Emacs permet également d'utiliser les possibilités de BBDB, qui sert alors de carnet d'adresses.

L'intérêt est de compléter l'adresse de destination du mail à partir d'un mot-clef. Ceci est possible en utilisant le raccourci `M-TAB` après avoir saisi une chaîne de caractères dans le champ `"To:"` de la composition. BBDB consulte alors tous les champs de sa base de données, et dans le cas où un seul champ complète la chaîne, le champ net de l'enregistrement concerné remplace la chaîne saisie. En cas d'ambiguïté, un buffer propose les différentes solutions possibles.

2.2 Comment lire les news ?

Gnus est un lecteur de news écrit en *Emacs-lisp*. Il est conçu pour lire les forums d'*Usenet*, pour y poster et même pour lire son mail. Gnus s'exécute par `M-x gnus`, ce qui fait apparaître une splendide image de Gnu.

2.2.1 Introduction

Pour lire les news, Gnus ouvre plusieurs buffers:

- le buffer `*Group*`: il permet de naviguer à travers les groupes de news. Il permet aussi de trier les groupes et de les manipuler: souscription, ...
- le buffer `*Summary*`: il contient le sommaire de tous les articles contenus dans un groupe de news. Il est créé lors de la consultation d'un groupe de news.
- le buffer `*Article*`: il affiche un article, et est créé par une commande effectuée dans le buffer `*Summary*`.

2.2.2 Utilisation

Gnus lit au démarrage le fichier `.newsrc` du répertoire racine de l'utilisateur. Ce fichier permet de savoir à quels groupes il a souscrit, et lesquels il a décidé de ne pas lire (groupes *killed*).

La lecture de news se fait en deux étapes:

- la sélection de groupes dans le buffer `*Group*`. Elle permet de consulter les différents groupes disponibles sur le serveur de news.
- la sélection d'articles dans le buffer `*Summary*`. Chaque article est contenu par un groupe d'articles (ou *thread*), organisé selon les réponses que l'article a reçu.

Commandes du buffer `*Group*`

- `q`: quitte le mode *gnus*, et met à jour le fichier `.newsrc`
- `L`: liste tous les groupes, sauf ceux qui ont été marqués comme étant à ignorer (*killed*) par l'utilisateur
- `l`: liste tous les groupes auxquels a souscrit l'utilisateur et qui contiennent des articles non encore lus
- `u`: souscrit ou annule la souscription (selon le cas) du groupe sélectionné
- `SPC`: crée un buffer `*Summary*` qui liste tous les articles du groupe sélectionné

Commandes du buffer `*Summary*` Notons que toutes ces commandes peuvent aussi s'exécuter dans le buffer `*Article*`.

- `q`: quitte le buffer `*Summary*` pour retourner sur le buffer `*Group*`. Utiliser `q` une deuxième fois quitte *gnus*
- `n`: déplace la sélection sur le prochain article non lu
- `p`: déplace la sélection sur l'article non lu précédent
- `SPC`: fait défiler le buffer `*Article*` d'une page vers le bas, et sélectionne l'article suivant si le buffer `*Article*` est à la fin
- `DEL`: fait défiler le buffer `*Article*` d'une page vers le haut, et sélectionne l'article précédent si le buffer `*Article*` est déjà au début
- `M-s` *regexp*: recherche dans le buffer `*Summary*` des articles contenant l'expression régulière *regexp*

Commandes de postage d'article

- `a`: poste un nouvel article, par défaut dans le groupe courant
- `f`: poste un article en réponse à l'article sélectionné
- `F`: poste un article en réponse à l'article sélectionné, et insère l'article en question précédé par des ">" (ce signe est réglable par la variable *mail-yank-prefix*)
- `r`: répond par mail à l'auteur de l'article sélectionné
- `R`: répond par mail à l'article sélectionné, et insère l'article en question précédé par des "> " (ce signe est réglable par la variable *mail-yank-prefix*)

2.3 Comment puis-je accéder à un site FTP ?

FTP est un protocole permettant le transfert de fichiers à travers Internet entre un *client* et un *serveur*. Il permet par le biais d'un client et d'un jeu de commandes réduit d'accéder à un disque distant. Les commandes de base sont `ls`, `cd`, `chmod`, et des commandes supplémentaires permettent de transférer des fichiers entre le client et le serveur.

Ange-FTP est un mode d'Emacs qui permet d'utiliser FTP d'une manière transparente. Sa puissance est de pouvoir être couplé à l'utilisation de Dired (voir section 4.3 (Dired) pour plus de détails). En fait, Ange-FTP se contente de se *logger* sur un serveur FTP en utilisant le nom de l'utilisateur, et en effectuant une liste des fichiers présents, qu'il passe ensuite à dired.

Ange-FTP est invoqué en ouvrant un fichier dont le nom suit une syntaxe spéciale, de quelque manière que ce soit (ouverture, enregistrement, ...). Son utilisation est transparente : on peut très bien éditer les fichiers d'un site web à partir de sa station, ou utiliser dired en conjonction avec Ange-FTP pour manipuler facilement une arborescence.

La syntaxe reconnue par Ange-FTP est /*login*@*site*:*chemin*. Lorsqu'un fichier sera ouvert ou sauvegardé sous un nom suivant cette syntaxe, alors Ange-FTP entrera en action, et rapatriera le fichier désigné ou mettra à jour le serveur.

Par exemple, pour ouvrir un buffer en mode dired sur le site `ftp.lip6.fr` afin de lister le répertoire `/pub/emacs/elisp`, il suffit d'ouvrir `/ftp@ftp.lip6.fr:/pub/emacs/elisp/`.

Notons que le mode Ange-FTP bloque totalement Emacs lors du téléchargement d'un fichier. Le mode EFS, présenté comme le successeur de Ange-FTP (A faire...) permettra de corriger cet inconvénient.

2.4 Comment naviguer sur le web ?

Emacs fournit un navigateur Internet appelé w3-Emacs. Dans sa version texte, il ressemble quelque peu à Lynx. Dans sa version XEmacs les images sont incluses et le résultat est quelque peu similaire à celui de Mosaic. w3-Emacs est certes textuel, mais il reconnaît les tables et les frames.

W3-emacs est un énorme programme. Sa configuration est déroutante si on n'est pas habitué à *Emacs-Lisp*. C'est pourquoi je recommande à toute personne désirant en savoir plus de visiter le site `http://www.cs.ucsb.edu/Facilities/Software/xemacs-info/w3/Top.html` .

Une fois que w3-Emacs est installé et configuré correctement, son utilisation est simple. Son appel se fait par l'intermédiaire de la commande `w3`. La navigation d'un lien à un autre se fait par l'intermédiaire de la touche `TAB` (lien suivant) ou `S-TAB` (lien précédent). Cliquer avec le bouton du milieu de la souris sur un lien hypertexte permet de le suivre et d'afficher la page HTML liée dans un nouveau buffer.

Principales commandes de w3 :

- `C-o` : demande une URL et ouvre la page correspondante dans un nouveau buffer
- `DEL` : fait défiler la page d'un écran vers le haut
- `SPC` : fait défiler la page d'un écran vers le bas
- `B` : retourne à l'URL précédente dans la navigation
- `F` : passe à l'URL suivante dans la navigation
- `v` : affiche l'URL du document dans la zone d'écho
- `s` : affiche le source du document dans un autre buffer
- `r` : recharge le document
- `d` : télécharge ce document
- `q` : quitte le buffer

Il existe bien d'autres commandes dans le mode w3 (notamment la gestion des bookmarks, de l'historique, ...). Je conseille à toute personne intéressée de consulter la documentation en ligne Info d'Emacs.

3 Développement

3.1 Comment compiler sans basculer dans un terminal ?

La compilation de programmes s'effectue généralement par l'intermédiaire du programme make (voir `http://www.april.org/groupes/doc/make/make.html`).

Emacs fournit une interface de compilation, qui s'exécute par le biais de la commande compile (`M-x compile`). Une fenêtre contenant le buffer `*compilation*` apparaît alors, et affiche la sortie standard de la compilation. Il peut être utile de créer un raccourci pour compile au moyen de local-set-key ou de global-set-key (j'utilise personnellement `C-c RET`).

L'intérêt de cette interface est la gestion des erreurs. Lorsque la compilation ne réussit pas, une convention est que le compilateur affiche un message d'erreur du type :

```
client.c:33: parse error before '{'
```

Emacs analyse cette ligne, et peut en déduire qu'il y a une erreur dans le fichier `client.c` à la ligne 33. Les différentes erreurs sont répertoriées, ce qui autorise alors le mode compilation à proposer une navigation dans celles-ci ainsi que des liens vers les fichiers concernés.

Le mode compilation possède les raccourcis suivants :

- `ESC {` : permet de déplacer le *point* sur le fichier précédent contenant une erreur
- `ESC }` : permet de déplacer le *point* sur le prochain fichier contenant une erreur
- `ESC p` : permet de déplacer le point sur l'erreur précédente
- `ESC n` : permet de déplacer le point sur l'erreur suivante
- `C-c C-k` : arrête la compilation
- `C-c C-c` ou `RET` : ouvre le fichier se trouvant sous le point dans une nouvelle fenêtre, et déplace le point dans cette fenêtre à la ligne fautive

3.2 Est-ce possible de compiler un document LaTeX sans douleur ?

Emacs fournit bien sûr un mode LaTeX, et une des possibilités de ce mode est la compilation du fichier édité : Emacs permet de lancer un processus de compilation sur le buffer courant ou sur une région.

Commandes LaTeX :

- `C-c C-b` : invoque LaTeX sur le *buffer* courant
- `C-c TAB` : invoque BibTex sur le *fichier* courant
- `C-c C-f` : invoque LaTeX sur le *fichier* courant
- `C-c C-k` : arrête le processus de compilation LaTeX
- `C-c C-v` : visualise le fichier de sortie de la dernière commande de compilation LaTeX (`xdvi`, spécifié par la variable `tex-dvi-view-command`)
- `C-c C-p` : imprime le fichier de sortie de la dernière commande de compilation LaTeX (`lpr`, spécifié par la variable `tex-dvi-print-command`)
- `C-c C-q` : affiche la queue d'impression (`lpq`)

Afin de faciliter l'édition de documents, le mode LaTeX propose deux raccourcis pour insérer les commandes LaTeX begin (`C-c C-o`) et end (`C-c C-e`).

3.3 Où trouver un interpréteur Lisp ?

3.3.1 Emacs-Lisp

Emacs est un éditeur de textes basé sur le langage Lisp. Il est donc normal qu'il fournisse une excellente interface de développement Lisp. Le langage utilisé est un dialecte de Common-Lisp : l'Emacs-Lisp.

Emacs fournit un mode majeur pour éditer toutes sortes de codes Lisp, mais il possède également un mode Emacs-Lisp, qui propose de plus l'évaluation d'expressions (*sexps*).

Il existe plusieurs façons d'utiliser ce mode :

- utiliser `M-x ielm` (Interactive Emacs Lisp Mode), qui crée un buffer `*ielm*`, où l'utilisateur peut entrer interactivement des instructions Emacs-Lisp qui seront évaluées au fur et à mesure de leur saisie.
- basculer dans le buffer `*scratch*`, qui est normalement créé au démarrage de toute session Emacs, et qui est par défaut en mode Emacs-Lisp.
- ouvrir ou créer un fichier dont l'extension est `.el`, ce qui fait basculer le buffer en mode Emacs-Lisp. Ceci est spécifié par la variable `auto-mode-alist`.

Emacs permet d'évaluer des expressions Lisp de plusieurs manières :

- `M-x eval-buffer` permet d'évaluer le buffer tout entier. C'est un bon moyen de charger un fichier Emacs-Lisp.
- `M-C-x` permet d'évaluer l'expression située sur le point ou juste avant. Avec un argument, cette commande imprime le résultat dans le buffer courant au niveau du point.
- `C-x C-e` permet d'évaluer la dernière expression Lisp. Avec un argument, cette commande imprime le résultat dans le buffer courant au niveau du point.

3.3.2 Autres Lisps

Emacs offre des interfaces avec d'autres lisps.

A faire...

3.4 Comment utiliser le contrôle de version ?

VC (Version Control) vous permet d'utiliser un système de contrôle de version directement sous Emacs. VC fournit une interface uniforme quelque soit le système de contrôle de version qui est utilisé (RCS, CVS et SCCS). Les commandes de VC sont conceptuellement proches des commandes RCS, ainsi presque tout ce que vous pouvez faire avec RCS peut être fait à travers VC.

RCS utilise normalement le *locking*, c'est-à-dire qu'une seule personne peut modifier un fichier. Pour cela, les fichiers de travail sont en mode lecture seule. Si vous souhaitez apporter des modifications et les intégrer au *master* (le fichier qui contient l'istorique des modifications), vous devez demander au système de contrôle de rendre le fichier de travail accessible en écriture pour vous et le vérouiller. Un seul et unique utilisateur peut faire cela à un instant donné. Lorsque vous validerez vos modifications (par un checkin), le fichier sera dévérouillé, et le fichier de travail de nouveau en lecture seule.

Une fois que vous avez créé un fichier, vous devez l'enregistrer dans le système de contrôle de version par la commande `M-x vc-register` (`C-x v i`) qui crééra le fichier master (dont le nom se termine par `,v`) correspondant au fichier édité. Le fichier master représente l'état actuel du fichier plus tous les historiques de changement, ainsi que les entrées de journalisation decrivant en quelques mots les modifications apportées pour chaque version du fichier. Ainsi, si vous éditez le fichier foo.c, la commande `vc-register` crééra le fichier foo.c,v.

Ensuite, chaque fois que vous visiterez un fichier qui est sous le sytème de contrôle, Emacs l'indiquera par exemple avec RCS-1.2 dans la fenêtre d'état (signalant ainsi que la version du fichier est la 1.2). Le caractère entre RCS et le numéro de version indique le status du fichier. '`-`' signifie que le fichier n'est pas vérouillé (si le locking est utilisé) ou n'est pas modifié (si le locking n'est pas utilisé). '`:`' indique que le fichier est vérouillé, ou qu'il est modifié. Si le fichier est vérouillé par un autre utilisateur (par ex, `benj`), c'est indiqué comme suit '`RCS:benj:1.2`'.

Si vous avez créé un fichier, enregistré sous le système de contrôle et que vous voulez y apporter des modifications, vous ouvrez donc le fichier (`C-x C-f`). Le fichier de travail étant en lecture seule, vous ne pouvez pas le modifier. Vous devez d'abord vérouiller le fichier (`M-x vc-toggle-read-only`, `C-x C-q`). Vous apportez des modifications, et une fois que vous voulez les intégrer dans le master, vous utilisez la commande `vc-next-action` (`C-x v v`). Celle-ci ouvrira un buffer pour que vous puissez entrer le message de journalisation (*log*) indiquant en quelques mots les raisons des modifications. Pour valider, tapez `C-c C-c` (`M-x compile`).

En fait, la commande `C-x v v` peut suffire pour la plupart des opérations, une fois un fichier enregistré auprès du système de contrôle. L'action entrainée par cette commande dépendra de l'état du fichier et du fait que le système de contrôle de version utilise le locking ou non.

Pour une documentation sur le contrôle de version indépendant d'Emacs, voir la page du groupe de documentation APRIL (`http://www.april.org/Travaux/Doc/`)

4 Divers

4.1 Mode `tar` et décompression à la volée

A faire...

4.2 Et pour écouter des CD audios ?

Il est possible d'écouter des CD audios en utilisant le programme workbone, qui peut-être interfacé avec Emacs par le biais d'un petit programme Emacs-Lisp disponible sur `http://inferno.cs.univ-paris8.fr/~drieu/hacks/workbone.el` . Cette interface s'exécute par le biais de la fonction workbone (`M-x workbone`).

Les commandes sont :

- `C-x w p` *n* : joue la piste numéro *n*
- `C-x w +` *n* : augmente le son (nécessite l'installation du programme *volume*)
- `C-x w -` *n* : diminue le son (nécessite l'installation du programme *volume*)
- `C-x w 9` : exécute la lecture du CD (effectué automatiquement au démarrage)
- `C-x w 8` : effectue ou arrête une pause dans la lecture
- `C-x w 7` : arrête la lecture du CD
- `C-x w 6` : joue le morceau suivant
- `C-x w 5` : recommence le morceau courant
- `C-x w 4` : joue le morceau précédent
- `C-x w 3` : avance la lecture du morceau de 15 secondes
- `C-x w 2` : éjecte le CD
- `C-x w 1` : fait reculer la lecture de 15 secondes
- `C-x w 0` : quitte le processus sans arrêter la lecture

4.3 Puis-je manipuler les répertoires à la Midnight/Norton commander ?

Norton commander est un logiciel sous DOS permettant de remplacer à peu près un *shell*. Midnight Commander est un équivalent libre sous Unix de ce logiciel propriétaire. La plupart des opérations offertes par MC sont des opérations de manipulation de fichiers ou de répertoires.

Emacs permet d'effectuer ce type d'opérations par le biais du mode dired. Dired recense tous les fichiers d'un répertoire dans un buffer, et permet d'appliquer des opérations à chacun d'entre eux. La plupart des commandes du mode text sont disponibles dans ce buffer, plus des commandes propres au mode dired.

4.3.1 Appel de dired

Dired est invoqué par :

- `C-x d`, ou par `M-x dired`. L'utilisateur doit spécifier un répertoire à lister.
- l'*ouverture* d'un répertoire, par le biais de `C-x C-f` (find-file). Notons qu'Emacs préfère que l'utilisateur oublie le slash ('/') final. Si l'utilisateur l'inclut dans le nom du répertoire, Emacs ne créera pas de buffer accueillant le mode dired, mais basculera vers un buffer contenant un fichier se situant dans le répertoire s'il en existe un.

4.3.2 Marques

Les opérations sur les fichiers du mode dired se font soit sur le fichier courant (celui sur lequel est le *point*), soit sur un groupe de fichiers (ce qui nécessite de pouvoir marquer les fichiers sur lesquels les commandes vont agir).

Différentes commandes du mode dired permettent d'effectuer ce marquage :

- `*` : marque les fichiers exécutables
- `/` : marque les répertoires
- `@` : marque les liens symboliques
- `m` : marque le fichier courant
- `u` : retire la marque du fichier courant
- `DEL` : retire la marque du fichier précédent, et déplace le point sur ce fichier
- `M-DEL` : supprime les marques de tous les fichiers du répertoire
- `% m` *regexp* : marque tous les fichiers dont le nom *matche* une expression régulière

D'autres marques existent (fichier copié, fichier renommé, ...), et la plus utile à manipuler est la marque *à effacer* :

- `D` : permet de désigner le fichier courant comme fichier à effacer
- `#` : marque les fichiers de sauvegarde automatique d'Emacs (commençant par un *dièse*) comme fichiers à effacer
- `~` : marque les fichiers de backup d'Emacs (finissant par un *tilde*) comme fichiers à effacer
- `% d` : marque comme à effacer tous les fichiers dont le nom *matche* une expression régulière

4.3.3 Déplacement

- `n` : déplace le point sur le fichier suivant
- `p` : déplace le point sur le fichier précédent
- `M-{` : déplace le point sur le fichier marqué précédent
- `M-}` : déplace le point sur le fichier marqué suivant

4.3.4 Manipulation des fichiers

L'intérêt du mode dired est d'appliquer des commandes aux fichiers, dont la plupart peuvent remplacer avantageusement les commandes rentrées sous un *shell*. Toutes les commandes suivantes s'appliquent soit sur le fichier courant (sous le point) si aucun fichier n'est marqué, soit sur tous les fichiers marqués.

- `!` : demande puis exécute une commande *shell* sur les fichiers marqués. Cette commande (par exemple `"file"`) est exécutée sur tous les fichiers marqués, un par un, ou alors une seule fois sur tous les fichiers dans le cas où le caractère `*` se trouve dans la ligne de commande (ex: `"display *"`)
- `A` : cherche une expression régulière dans le contenu du/des fichier(s)
- `C` : effectue une copie de fichier(s)
- `D` : efface le(s) fichier(s)
- `H` : effectue un lien *hard* (dur)
- `M` : modifie les permissions
- `Q` : effectue un remplacement d'expressions régulières dans le contenu
- `R` : renomme ou déplace
- `S` : effectue un lien symbolique
- `Z` : compresse le(s) fichier(s) avec gzip ou les décompresse le cas échéant

4.3.5 Commandes diverses

- `RET` : ouvre le fichier courant dans un nouveau buffer
- `v` : ouvre en lecture seule le fichier courant (avec un mode rappelant un *pager*)
- `x` : efface les fichiers marqués comme à effacer
- `+` : crée un répertoire
- `^` : exécute dired sur le répertoire parent
- `s` : bascule entre le tri par date et le tri par nom
- `q` : fait disparaître le buffer dired

4.4 Est-ce qu'il existe un agenda ?

4.4.1 Calendar

Calendar est une interface permettant de manipuler un calendrier. Elle permet d'afficher un calendrier de trois mois dans un buffer, avec le point sur la date courante.

Parmi les actions possibles :

- insertion d'un rendez-vous pour une journée précise (voir section 4.4.2 (Diary))
- consultation des vacances pour une journée donnée
- impression en LaTeX d'un calendrier
- comptage de jours dans une période donnée

Le *point* désigne un jour, et l'utilisateur peut le déplacer à travers le temps par le biais de plusieurs commandes :

- avec les flèches : certains déplacements sont spéciaux (par exemple, se déplacer du dimanche au lundi ne correspond pas à un déplacement vers la droite, mais à un passage à la ligne)
- `M-}` : déplace le point d'un mois dans le futur
- `M-{` : déplace le point d'un mois dans le passé
- `C-x ]` : déplace le point d'un an dans le futur
- `C-x [` : déplace le point d'un an dans le passé
- `.` : replace le point sur aujourd'hui

Vacances

- `h` : affiche tout jour férié connu pour cette date
- `x` : affiche les jours fériés sur le calendrier
- `u` : désactive l'affichage des jours fériés sur le calendrier
- `a` : affiche les vacances des trois mois visibles dans un autre buffer

4.4.2 Diary

Diary est une interface permettant de gérer un agenda sous Emacs. Il s'utilise de concert avec Calendar qui permet d'insérer des entrées dans l'agenda pour des jours précis.

Le principe est qu'un fichier (dont le nom est renseigné par la variable `diary-file`) contient des entrées dans un format qui lui est propre (documenté dans le manuel Emacs). Des commandes permettent d'insérer des entrées dans l'agenda au format Diary à partir du buffer *Calendar*.

Parmi celles-ci :

- `d` : affiche toutes les entrées de l'agenda valides pour cette journée
- `s` : affiche toutes les entrées de l'agenda

- `i d` : ajoute une entrée pour la journée sélectionnée (sous le *point*)
- `i w` : ajoute une entrée pour le jour de la semaine sélectionné (ex: tous les mercredis, soirée gothique au *piano vache*)
- `i m` : ajoute une entrée pour le jour du mois sélectionné (ex: acheter carte orange tous les 1er du mois)
- `i y` : ajoute une entrée pour le jour de l'année sélectionné (ex: tous les 1er avril)
- `i b` : ajoute une entrée pour une période. L'utilisateur marque une date (`C-SPC`) puis déplace le curseur sur une autre date. La période entière sera ajoutée dans l'agenda (ex: du 29 juillet au 14 août, randonnée dans le pays Basque)
- `i c` : ajoute une entrée cyclique dans l'agenda. L'utilisateur sélectionne une date, puis insère une entrée cyclique. Diary lui demande alors une période en jours, et l'entrée sera insérée avec cette période dans l'agenda (ex: laver la voiture tous les 15 jours)

4.5 Comment imprimer un document texte ?

Sous la plupart des systèmes UNIX, la manière la plus simple d'imprimer un document est d'utiliser le PostScript. Emacs permet de formater des documents texte en PostScript, et éventuellement de les imprimer directement.

L'utilisation de quatre commandes permet d'effectuer cette impression :

- ps-print-buffer : convertit le buffer en PostScript et l'imprime. Le résultat sera une page de texte en police *courier* par défaut, avec une barre de titre en haut de chaque page, qui rappelle le nom du fichier, la date d'impression, etc ...
- ps-print-buffer-with-faces : cette fonction effectue le même travail que la précédente, mais elle conserve de plus les différentes polices appliquées à des caractères du buffer. Par exemple, un nom de fonction affiché en bleu sera imprimé en bleu sur une imprimante couleur
- ps-print-region : cette fonction imprime la région d'un buffer, tout comme l'effectue ps-print-buffer sur le buffer tout entier
- ps-print-region-with-faces : cette fonction imprime la région d'un buffer, tout comme l'effectue ps-print-buffer-with-faces sur le buffer tout entier

Il est possible de paramétrer la commande utilisée pour imprimer le fichier PostScript ainsi que les options passées à cette commande par l'intermédiaire des variables lpr-command (commande exécutée) et lpr-switches (liste de paramètres passés à cette commande). Par exemple, `(set-variable 'lpr-switches '("-Pche"))` (mon imprimante s'appelle che)

4.6 Il y a des jeux ?

4.6.1 Doctor

Doctor (`M-x doctor`) est un petit programme qui implémente le fameux robot *eliza*. Il s'agit d'une simulation de personne vivante : l'utilisateur pose des questions en langue naturelle au programme, et celui-ci lui répond par des phrases construites. Plus précisément, le but de Doctor est de simuler une consultation psychothérapique traditionnelle. Notons que la version actuelle de Doctor a été censurée par Richard Stallman lui-même en raison du côté "indécent" du code source (voir `http://www.april.org/groupes/gnufr/censoring-emacs.html`).

4.6.2 Getris

Mais oui, Tetris a été porté sous Emacs !

A faire...

4.6.3 BlackBox

BlackBox (`M-x blackbox`) est un jeu très simple. Le principe est de trouver des atomes cachés dans un boite noire en y envoyant des rayons lasers.

Lorsqu'un rayon touche un atome, il est absorbé par celui-ci. Lorsqu'il arrive à la diagonale d'un atome, il est dévié de 90 degrés de manière à s'éloigner de celui-ci. Par exemple :

```
                    O
                     +------->
        O            |
-------+       ------+
       |              O
       |
       v
```

Lorsque le jeu commence, une grille vide apparaît. Le but du jeu est de déplacer le curseur autour de la grille avec les flèches, et d'appuyer sur la barre d'espacement pour lancer un rayon laser. Un numéro apparaît, qui permet d'identifier le rayon. Si le rayon n'est pas absorbé, alors il est repéré par le même numéro en sortie. Si le rayon est absorbé, alors la lettre `H` (*hit*) est affichée à la place. Si par hasard le rayon sort au même endroit d'où il est parti, alors la lettre `R` (*reflected*) est affichée à la place.

Quand le joueur pense avoir deviné où se trouve un atome, alors il place le curseur sur la grille, et appuie sur la barre d'espacement pour placer un atome. Quand il a placé tous les atomes, il appuie sur la touche `RET` afin de valider, ce qui lui permet de voir la position réelle des atomes.

4.7 Dessiner en ASCII

Emacs offre deux modes pratiques pour le dessin ASCII.

4.7.1 le mode `picture`

A faire...

4.7.2 le mode `artist`

A faire...

4.8 Comment vérifier l'orthographe d'un texte ?

4.8.1 Ispell

Un des correcteurs orthographiques les plus populaires sous UNIX est ispell. Emacs intègre logiquement une interface permettant de le manipuler sur un buffer.

Le principe est qu'Emacs lance un processus ispell, et ensuite qu'il lui envoie des données texte à partir du buffer. A chaque mot incorrect, Emacs propose un choix parmi plusieurs possibilités par l'intermédiaire d'un autre buffer. Il est possible de lancer un processus ispell sur plusieurs parties d'un buffer :

- `M-$` : exécute ispell sur le mot se trouvant sous le point
- `M-TAB` : complète le mot sous le point par le premier mot correct parmis ceux proposés par le dictionnaire
- `M-x ispell-buffer` : exécute ispell sur le buffer entier
- `M-x ispell-region` : exécute ispell sur la région

Pour répondre aux besoins de dictionnaires en plusieurs langues, ispell peut passer d'un dictionnaire à un autre, par le biais de la fonction ispell-change-dictionnary. L'utilisateur saisit un dictionnaire, et ensuite toute correction se fera dans cette langue (à condition que le dictionnaire soit installé sur le système).

4.8.2 Flyspell

Flyspell est une interface supplémentaire à ispell, qui permet d'utiliser ispell "à la volée" sur un texte, et de mettre en évidence un mot mal orthographié dès sa saisie.

Flyspell est un mode mineur. Il suffit donc de passer en mode flyspell (`M-x flyspell-mode` en considérant que flyspell est correctement installé) pour l'utiliser.

A chaque mot incorrectement orthographié, l'utilisateur a le choix entre utiliser les raccourcis d'ispell, et utiliser un menu contextuel invoqué par un clic avec le bouton du milieu de la souris. Ce menu propose un ensemble de mots corrects, ou propose à l'utilisateur de sauvegarder le mot dans le dictionnaire.

Flyspell est disponible sur `http://kaolin.unice.fr/~serrano/emacs/emacs.html` , mais est intégré aux nouvelles versions d'Emacs.

4.9 Comment on lance vi ?

Comme chacun le sait, la rivalité Emacs/vi fait rage.

Des développeurs Emacs on cependant essayé de trancher en proposant un mode dans lequel les raccourcis clavier sont ceux de vi, mais qui permet de profiter de toutes les facilités d'Emacs. Ce mode est le mode viper (**V**iper **I**s a **P**ackage for **E**macs **R**ebels).

Il est exécuté par `M-x viper-mode`, qui bascule alors en mode viper. Lorsqu'il est exécuté pour la première fois, ce mode affiche une page d'aide qui explique entre autres comment quitter ce mode.

L'utilisateur a ensuite le choix entre plusieurs niveaux d'utilisation (entre *beginner* et *wizard*), qui déterminent jusqu'à quelle profondeur vi sera émulé. Par exemple, aux niveaux les plus hauts, le raccourci `:wq` quittera Emacs et les touches d'Emacs seront remplacées par celles de vi.

`M-x viper-go-away` permet heureusement de quitter ce mode, et de revenir à un Emacs normal à tout moment.

5 Conclusion

Emacs est bien plus qu'un éditeur de textes. On l'a vu, c'est un environnement de travail complet (même s'il ne fait pas encore le café). Sa plus grande force (et peut-être aussi sa plus grande faiblesse) est d'être pratiquement entièrement adaptable pour le programmeur. En cas de besoin, il est très facile d'implémenter la fonction de ses rêves (recherche d'occurences particulières, traitement de mails, ...). Ce qui implique des talents de programmeur qui peuvent rebuter le novice, qui cherche simplement à adapter le logiciel à ses besoins.

5.1 Pointeurs

Deux sites proposant des programmes Emacs-Lisp prêts à l'emploi :

- l'Emacs-Lisp archive, sur le site FTP `archive.cis.ohio-state.edu`, dans (`ftp://archive.cis.ohio-state.edu/pub/emacs-lisp`) ;
- `http://www.emacs.org/`

Sur les news :

- `fr.comp.applications.emacs` : le newsgroup en français dédié à Emacs. Beaucoup de questions trouvent leur réponse ici
- `comp.emacs` : le même groupe, mais international
- `gnu.emacs.sources` : le groupe des développeurs d'Emacs. On y trouve beaucoup d'annonces pour des nouveaux modes en Emacs-Lisp.

6 Copyright