Gnu Emacs

Benjamin Drieu (drieu@bocal.cs.univ-paris8.fr) 10 Février 1998 - Version 2.0

Ce document est une brève introduction à **Gnu Emacs** Si vous voulez en savoir plus sur cet éditeur, je vous conseille de vous reporter à la documentation officielle d'Emacs que vous trouverez sur le site FTP de GNU (ftp://prep.ai.mit.edu/pub/gnu), ou à sa traduction en cours par l'association APRIL. Si vous aussi voulez participer à la traduction du manuel d'Emacs et au formidable élan du Free Software, rendez vous sur le site web d'APRIL (http://www.april.org) pour y trouver plus d'informations.

Contents

1 Introduction **2**

1.1 Qu'est-ce qu'Emacs ? . . . 2

1.2 Conventions utilisées . . . 3

2 Buffer, Fichier, Fenêtre **3**

2.1 Manipulation de fichiers . . . 3

2.2 Manipulation de buffers . . . 4

2.3 Manipulation de fenêtres . . . 4

3 Organisation de l'écran **4**

3.1 Barre de menu . . . 4

3.2 Fenêtre . . . 4

3.3 Ligne de mode . . . 5

3.4 Ligne d'écho . . . 5

4 Interaction avec Emacs **5**

4.1 Point . . . 5

4.2 Saisie . . . 5

4.3 Déplacement . . . 5

4.4 Marque et région . . . 6

4.5 Commandes, arguments . . . 6

5 Kill-ring et registres - copie de texte **7**

5.1 Kill-ring . . . 7

5.2 Registres . . . 7

6 Recherche et remplacement **8**

6.1 Recherche . . . 8

6.1.1 Recherche incrémentale . . . 8

6.1.2 Recherche non-incrémentale 8
6.1.3 Recherche d'expressions régulières 8
6.2 Remplacement 9
6.2.1 Remplacement inconditionnel 9
6.2.2 Remplacement conditionnel 9

7 Modes **10**

8 Aide et documentation **10**
8.1 Manuel *Info* 10
8.2 Aide contextuelle 11

9 Applications **12**
9.1 RMail 12
9.1.1 Introduction 12
9.1.2 Mails 12
9.1.3 Listes de mails 12
9.1.4 Commandes d'envoi de mail 13
9.1.5 Manipulations sur fichiers RMail 13
9.1.6 Variables de RMail 13
9.2 Gnus 13
9.2.1 Introduction 14
9.2.2 Utilisation 14
9.3 Divers 15

1 Introduction

1.1 Qu'est-ce qu'Emacs ?

Emacs est un éditeur de textes plein écran interractif en temps réel. Emacs signifie *Editor MACroS*, et à été développé par Richard Matthew 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 majeur 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 permet à l'utilisateur d'apprendre Emacs sans imprimer de manuel, et facilite l'apprentissage.

Emacs est aussi bien plus qu'un éditeur puisqu'il regorge de modes divers et d'applications annexes qui facilitent son utilisation. Parmis ces applications, certaines peuvent même remplacer des applications utilisées couramment, telles que les lecteurs de mails ou de news. Emacs offre aussi quelques jeux pour ce détendre, et même un psychothérapeuthe !!!

1.2 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 (notons que `C-X` produit le même effet).
- `M-x`: frappe différée de la touche `Esc` puis de la touche x.
- `M-x toto`: frappe différée de la touche `Esc` puis de la touche `x` puis de la séquence de caractères `toto`.
- `S-right`: frappe simultanée de la touche `shift` et du bouton droit de la souris.
- `S-C-right`: frappe simultanée de la touche `shift`, de la touche `Ctrl`, et du bouton droit de la souris.
- *chaîne*: une chaîne de caractères
- *regexp*: une expression régulière (en anglais, *regular expression*)

2 Buffer, Fichier, Fenêtre

Contrairement à certains éditeurs, Emacs n'édite pas de *fichiers*. Emacs édite des *buffers*. Un fichier est une suite d'octets sur un support quelconque, mais pour Emacs un buffer est une copie de travail de ce fichier. Toutes les modifications apportées à un texte sont en fait apportées à un buffer avant d'être enregistrées dans un fichier.

Emacs possède de plus une notion de *fenêtre*. Une fenêtre est un espace sur l'écran qui affiche un buffer. Elle contient de plus une ligne de mode qui donne des indications sur le buffer en cours d'édition (voir chapitre *Organisation de l'écran*).

Lorsque l'utilisateur saisit du texte, c'est dans la fenêtre qu'il le saisit, mais c'est dans le buffer que sont faites ses modifications. Lorsque l'utilisateur change de buffer la fenêtre reste la même mais le buffer affiché change.

2.1 Manipulation de fichiers

- `C-x C-f` *nom*: ouverture du fichier *nom* dans la fenêtre courante. Le buffer édité est mis en arrière plan, et reste dans la liste de buffers. Un nouveau buffer est créé
- `C-x C-v` *nom*: ouverture du fichier *nom* dans la fenêtre courante à la place du buffer édité. Le buffer édité est fermé (avec confirmation au besoin), et le fichier passé en argument est ouvert dans un buffer qui prendra sa place
- `C-x C-s`: enregistre le contenu du buffer courant dans le fichier sur lequel il est ouvert
- `C-x s`: enregistre le contenu de tous les buffers ouverts
- `C-x C-w` *nom*: enregistre le contenu du buffer édité sous le nom *nom*

2.2 Manipulation de buffers

- `C-x b` *buffer*: sélectionne ou crée (s'il n'existe pas) le buffer *buffer* dans la fenêtre courante
- `C-x 4 b` *buffer*: sélectionne ou crée le buffer *buffer* dans une nouvelle fenêtre
- `C-x C-b`: ouvre une nouvelle fenêtre contenant un la liste de tous les buffers ouverts. Je conseille à toute personne intéressée de se reporter à la section *Listing Existing Buffers* du manuel Emacs pour plus d'informations au sujet de cette liste et comment l'utiliser
- `M-x rename-buffer`: renomme le buffer courant (notons que cela n'affecte pas le nom du fichier visité)
- `C-x k` *buffer*: détruit le buffer *buffer*, ou le buffer courant si aucun buffer n'est passé en argument

2.3 Manipulation de fenêtres

- `C-x 2`: Découpe verticalement la fenêtre courante en deux fenêtres
- `C-x 3`: Découpe horizontalement la fenêtre courante en deux fenêtres
- `C-x o`: Sélectionne la fenêtre suivante
- `C-x 0`: Détruit la fenêtre sélectionnée
- `C-x 1`: Détruit toutes les fenêtres sauf la fenêtre courante
- `C-x +`: Réorganise toutes les fenêtres, et leur donne une taille égale

3 Organisation de l'écran

L'écran d'Emacs est découpé au démarrage en quatre zones distinctes:

- la barre de menu (*menubar*),
- la fenêtre (*window*),
- la ligne de mode (*mode line*),
- la zone d'écho (*echo area / minibuffer*)

3.1 Barre de menu

La barre de menu contient un certain nombre de raccourcis vers des commandes utiles. Dans un système avec pointeur (XWindow par exemple), l'utilisateur peut sélectionner un de ces raccourcis avec le pointeur. Dans un système sans pointeur les commandes `` M-` `` et `f10` permettent d'accéder à un mode où les commandes du menu peuvent être exécutées au clavier.

3.2 Fenêtre

La fenêtre contient le buffer en cours d'édition. Voir le chapitre *Buffers, Fenêtres, Fichiers* pour plus d'informations sur les fenêtres.

3.3 Ligne de mode

Elle contient des informations sur le buffer édité dans la fenêtre à laquelle elle appartient: le nom du buffer, les modes en cours, si le buffer contient des modifications non sauvegardées, et la position de la fenêtre dans le buffer.

3.4 Ligne d'écho

C'est la ligne qui se trouve en bas de la fenêtre. Elle est utilisée pour afficher le texte que vous saisissez lors d'une commande (*minibuffer*). Elle affiche de plus les messages d'erreur ainsi que tout le texte qu'Emacs renvoie lors d'une commande. Tous les messages affichés dans la ligne d'écho sont copiés dans un buffer (voir chapître *Buffer, Fichier, Fenêtre*) nommé `*Messages*`.

4 Interaction avec Emacs

L'interaction avec Emacs se fait soit par la saisie de texte dans la fenêtre, soit par l'appel de commandes dans la zone d'écho, qui se confond pour l'occasion avec le minibuffer. L'utilisateur peut y saisir des commandes ou les arguments des commandes appelées. La saisie de texte dans un buffer est effectuée au niveau du **point**.

4.1 Point

Il y a dans chaque buffer d'Emacs un endroit particulier où les commandes d'édition prennent effet, et où la saisie de texte est insérée. Il correspond au curseur du terminal, et est appelé *point* pour des raisons historiques. Le curseur du terminal est repéré par un caractère en inverse vidéo, mais le point se trouve en fait juste entre ce caractère et le précédent. Il se déplace bien entendu avec les flèches, mais aussi avec d'autres commandes de déplacement (voir la sections *Déplacement* pour plus d'informations au sujet du déplacement du point).

4.2 Saisie

La saisie sous Emacs peut se faire dans deux modes: *insertion* ou *overwrite*. Dans le mode insertion, la saisie de caractères se fait au niveau du point. Chaque caractère saisi est inséré au niveau du point, c'est à dire juste avant le curseur du terminal.

Dans le cas du mode *insertion*, chaque caractère saisi est inséré dans le texte. Mais dans le cas du mode *overwrite*, chaque caractère saisi remplace le caractère se trouvant juste après le point, c'est-à-dire juste sous le curseur du terminal.

4.3 Déplacement

Emacs possède de nombreuses commandes de déplacement. Les plus simples déplacent le point d'un caractère dans une direction, mais d'autres sont spécifiques à un *mode* précis (déplacement au *defun* précédent en mode lisp, par exemple).

- `C-a`: déplace le point au début de la ligne courante
- `C-e`: déplace le point à la fin de la ligne courante. Le curseur se trouve donc après les dernier caractère de la ligne

- `M-a`: déplace le point au début du paragraphe courant
- `M-e`: déplace le point à la fin du paragraphe courant
- `M-f`: déplace le point au début du mot courant
- `M-b`: déplace le point à la fin du mot courant
- `M-<`: déplace le point au début du buffer
- `M->`: déplace le point à la fin du buffer

4.4 Marque et région

En plus du point, chaque buffer d'Emacs possède une position nommée *marque* (*mark* en anglais). La marque permet de définir une *région*. Chaque buffer possède une région, qui lui permet d'exécuter des commandes sur une partie seulement du texte, par exemple sa copie, sa conversion en majuscules ou minuscules, ... La région du buffer est l'espace situé entre le point et la marque. On peut la modifier tout simplement en déplaçant le point ou la marque. Le point étant le curseur où le texte est inséré, il suffit de se déplacer. La marque est déplacée par l'utilisation de commandes spécifiques:

- `C-SPC`: Fixe la marque au niveau du point
- `C-@`: Même chose que la commande précédente, hormis que celle-ci fonctionne sur tout type de terminal
- `C-x C-x`: Echange le point et la marque: le curseur se déplace au niveau de la marque, et vice-versa. Cette commande est utile pour revenir à un endroit précis du buffer après des déplacements
- `M-@`: Fixe la marque à la fin du mot suivant (ou courant si le point est au milieu d'un mot). Le point n'est pas déplacé
- `M-h`: Fixe la région autour du paragraphe courant (déplace le point au début, et la marque à la fin)
- `C-x h`: Fixe la région autour du buffer entier
- `C-u C-SPC`: Restaure la marque précédente. Le point est déplacé au niveau de la marque, et la marque est déplacée au niveau de la marque précédente.

4.5 Commandes, arguments

Une commande s'exécute généralement sous Emacs par `M-x` *nom de la commande*. Il s'agit alors d'une fonction *Lisp*. Emacs demandera parfois à l'utilisateur de saisir des arguments dans le minibuffer. Mais il existe pour certaines de ces fonctions des raccourcis clavier afin d'exécuter la commande plus rapidement (par exemple, la commande `find-file` peut s'exécuter par `C-x C-f`).

L'utilisateur peut aussi donner des arguments numériques à une commande par le biais de la séquence de caractères `C-u` *argument*. Dans la plupart des cas, spécifier un argument de cette manière exécutera la commande plusieurs fois. Par exemple, `C-u 2 M-x downcase-word` exécute deux fois `downcase-word`, et convertit en minuscules les deux mots suivant le point (notons que `C-u 2 M-l` produit le même effet puisque `M-l` est un raccourci pour `downcase-word`. On peut obtenir le même résultat avec `M-`n, qui équivaut à un `C-u`n.

5 Kill-ring et registres - copie de texte

5.1 Kill-ring

La copie et le collage de texte sont assurés dans Emacs par les opérations au noms barbares de "*killing*" et de "*yanking*". Le principe est que l'utilisateur peut détruire du texte (*killing*), et le coller à partir du *kill-ring* dans un buffer.

Le kill-ring est un espace commun à tous les buffers où Emacs range les textes détruits précédemment. On peut le considérer comme une pile circulaire. A chaque fois que l'utilisateur effectue une destruction de texte, le texte en question est rangé à la fin du kill-ring. Lorsqu'une commande de *yanking* sera exécutée, ce texte sera inséré dans le buffer au niveau du *point*. Le kill-ring étant commun à tous les buffers, cela en fait un outil pratique pour déplacer du texte d'un buffer à un autre.

Commandes de destruction de texte:

- `C-k` : détruit la ligne courante à partir du *point*
- `C-w` : détruit la région (du point jusqu'à la *marque*)
- `M-z` *caractère*: détruit le texte jusqu'à la première occurrence du caractère *caractère*
- `M-DEL` : Détruit le mot précédent (ou le mot courant si le curseur est au milieu d'un mot)
- `M-d` : Détruit le mot suivant (ou le mot courant si le curseur est au milieu d'un mot)

Commandes de "collage" de texte et de manipulation de *kill-ring*:

- `C-y` : insère le bloc de texte situé à la fin du *kill-ring* au niveau du point. Avec un argument numérique, cette fonction insère le bloc précédent spécifié par l'argument. Par exemple: `C-u 2 C-y` insère l'avant-dernier bloc de texte détruit.
- `M-y` : juste après un `C-y`, remplace le texte inséré par le bloc de texte précédent dans le *kill-ring*.
- `M-w` : enregistre la région dans le *kill-ring* sans la détruire.
- `C-M-w` : si la commande suivante est une destruction de texte, alors le texte détruit sera ajouté à la fin du dernier bloc de texte enregistré dans le *kill-ring*.

5.2 Registres

Les registres d'Emacs sont des espaces où l'utilisateur peut enregistrer du texte ou une position, et les rappeler plus tard. Chaque registre est repéré par une lettre qui permet d'y accéder ultérieurement. L'utilisation des registres pour conserver du texte est quelque peu analogue à celle des buffers. Mais les registres permettent aussi de revenir à une position précise dans un buffer en sauvegardant une position.

Commandes d'utilisation de buffers:

- `C-x r SPC` *r*: sauvegarde la position du point dans le registre *r*
- `C-x r j` *r*: déplace le point à la position enregistrée dans le registre *r*
- `C-x r s` *r*: copie la région dans le registre *r*
- `C-x r i` *r*: insère le texte contenu dans le registre *r*

Une autre utilisation des registres est la sauvegarde des noms de fichiers. Elle se fait par l'intermédiaire d'un code Lisp, et les fichiers peuvent être rappelés par l'intermédiaire de la commande `C-x r j` *r*. Cette commande ouvre le fichier spécifié par le registre *r*. Le code Lisp correspondant est:

```
(set-register ?r '(file . nom))
```

Notons que la commande pour ouvrir un fichier dont le nom est obtenu par l'intermédiaire du registre est la même que celle pour déplacer le point à une position spécifiée par un registre: `C-x r j r`.

6 Recherche et remplacement

6.1 Recherche

6.1.1 Recherche incrémentale

La recherche de texte sous Emacs (*incremental search*) est légèrement différente de celle que font la plupart des autres éditeurs. Sa particularité est qu'elle est *incrémentale*, c'est-à-dire qu'elle commence à rechercher du texte avant même que l'utilisateur ai fini de le taper. L'utilisateur voit ainsi la recherche se faire et s'affiner au fur et à mesure qu'il saisit une chaîne. Ce type de recherche est pénible sur les terminaux à rafraîchissement lent (ou par telnet), mais il existe d'autres fonctions de recherche plus traditionnelles qui ne réaffichent pas l'écran à chaque saisie de caractère. L'*incremental search* commence au niveau du point, et peut continuer au début du buffer si elle arrive à sa fin.

Commandes de recherche incrémentale de texte:

- `C-s` *chaîne*: recherche incrémentale en avant
- `C-r` *chaîne*: recherche incrémentale en arrière

Notons que si plusieurs occurrences d'un même texte se trouvent dans un buffer, alors la recherche s'arrête à la première. L'utilisateur peut accéder aux autres occurrences en utilisant à nouveau la commande `C-s` (ou `C-r` pour une recherche en arrière). La commande `C-s C-s` permet de faire une recherche sur la dernière occurrence recherchée.

6.1.2 Recherche non-incrémentale

Emacs peut aussi rechercher une chaîne sans faire de recherche incrémentale, c'est à dire que le buffer sur lequel se fait la recherche ne subbira pas de raffraîchissement à chaque saisie de caractère.

- `C-s RET` *chaîne*: recherche la chaîne *chaîne* de manière non-incrémentale
- `C-r RET` *chaîne*: recherche la chaîne *chaîne* en arrière de manière non-incrémentale

6.1.3 Recherche d'expressions régulières

Une expression régulière est une chaîne de caractères contenant des "*caractères jokers*", qui permettent chacun de remplacer un ou plusieurs autres caractères, comme le jeton blanc permet de remplacer d'autres caractères au *scrabble*. La recherche d'expressions régulières est une recherche puissante qui permet de rechercher à peu près n'importe quoi: adresses électroniques, prix, groupes de mots, ...

La recherche d'expressions régulières se fait grâce à la commande `C-M-s` (recherche en avant) ou à la commande `C-M-r` (recherche en arrière). Leur utilisation est à peu de choses près la même que celle des commandes `C-s` et `C-r`. Le passage d'une occurrence à une autre se fait par l'utilisation des commandes `C-s` et `C-r` respectivement.

Par exemple:

```
C-M-s [-a-zA-Z_\.]+@[-a-zA-Z0-9_\.]+\.\(fr\|com\|net\|org\)
```

permet de rechercher la plupart des adresses électroniques.

Je conseille à qui voudrait en savoir plus sur les expressions régulières de se plonger dans le manuel d'Emacs à la section "*Syntax of Regular Expressions*" (environ cinq pages).

6.2 Remplacement

Il existe deux types de remplacement sous Emacs:

- le remplacement inconditionnel,
- le remplacement conditionnel

Chacun de ces remplacements a deux variantes: le remplacement de chaînes, et le remplacement d'expressions régulières.

6.2.1 Remplacement inconditionnel

Il remplace toutes les occurrences d'une chaîne passée en argument à partir du *point* par une autre chaîne passée elle aussi en argument sans demander confirmation à l'utilisateur.

Les commandes de remplacement sont:

- `M-x replace-string` *chaîne remplacement*: cette commande remplace sans confirmation toutes les occurrences de la chaîne *chaîne* par la chaîne *remplacement*.
- `M-x replace-regexp` *regexp remplacement*: cette commande remplace sans confirmation l'expression régulière *regexp* par la chaîne *remplacement*.

6.2.2 Remplacement conditionnel

La différence entre ce type de remplacement et le précéent est que le remplacement conditionnel demande confirmation à l'utilisateur pour chaque occurrence. Les commandes de remplacement conditionnel sont:

- `M-%` *chaîne remplacement*: cette commande remplace toutes les occurrences de la chaîne *chaîne* par la chaîne *remplacement* en demandant confirmation à l'utilisateur.
- `M-x query-replace` *chaîne remplacement*: cette fonction effectue la même tâche que la précédente
- `M-x query-replace-regexp` *chaîne remplacement*: cette commande remplace toutes les occurrences de l'expression régulière *regexp* par la chaîne *remplacement* en demandant confirmation à l'utilisateur.

A chaque fois qu'Emacs demande confirmation à l'utilisateur, celui-ci à le choix entre différentes possibilités, qu'il choisit en saisissant un caractère au clavier:

- `y`: remplace l'occurrence par la chaîne de remplacement et passe à la suivante
- `n`: ne modifie pas l'occurrence, mais passe à la suivante
- `.`: remplace l'occurrence par la chaîne de remplacement, mais arrête le remplacement sur cette occurrence
- `!`: remplace toutes les occurrences suivantes sans demander confirmation
- `^`: revient à l'occurrence précédente pour éventuellement y apporter des corrections

7 Modes

Une des particularités d'Emacs est l'utilisation des *modes*. Les *modes majeurs* d'Emacs modifient le comportement d'Emacs en fonction du type de texte édité. Cette modification de comportement permet de facilité la saisie de texte d'un mode ou d'un autre.

Chaque buffer possède un *mode majeur* et un seul. Les *modes mineurs* sont en revanche une option supplémentaire ajoutée au buffer. Un buffer peut ne pas posséder de mode mineur tout comme il peut en posséder plusieurs. Des exemples de modes mineurs sont les modes `auto-fill` (passage à la ligne automatique), le mode `auto-save` (sauvegarde automatique), `font-lock` (changement de polices dans le buffer), `iso-accents` (production de lettres européennes), `scroll-bar` (barre de défilement à coté de chaque buffer).

On peut passer d'un mode à l'autre en exécutant `M-x` *nom-du-mode*`-mode`.

Une variable d'Emacs permet de choisir quels modes seront utilisés par défaut lorsqu'un fichier est ouvert: `auto-mode-alist`. Elle est du format "`(`*regexp* `.` *mode*". Le mode *mode* est le mode exécuté lorsque l'expression régulière *regexp* est valide par rapport au nom du fichier ouvert. Par exemple, une ligne à ajouter dans le fichier de configuration d'Emacs.

```
(setq auto-mode-alist
      (append
      '(("\\.pov$" . pov-mode)
        ("\\.[c,h])$" . c-mode)
        ("\\.\\(c|cpp|cc\\)$" . c++-mode)
        ("\\.sgml$" . sgml-mode)
        ("^TO_DO" . change-log-mode))
      auto-mode-alist))
```

8 Aide et documentation

Un grand avantage d'Emacs est qu'il est **auto-documenté**. C'est-à-dire qu'il fournit lui-même de la documentation à l'utilisateur. Emacs possède deux types d'aide: le manuel *Info*, ou l'aide contextuelle.

8.1 Manuel *Info*

Le manuel Info est un manuel complêt, écrit au format `Texinfo`. Dans sa version imprimée, il fait a peu près 550 pages, mais le langage `Texinfo` permet une navigation hypertextuelle par le biais du mode `Info`. Le mode info permet la mise en page de fichiers *Info*, mais aussi l'insertion de liens cliquables afin de faciliter la navigation. Le manuel *Info* s'exécute par `C-h i`.

La philosophie du manuel *Info* est que toute documentation est organisée en arbre. Chaque section est un *noeud*. En tant que noeud, une section a un noeud précédent, un noeud suivant, ainsi qu'un noeud père et une racine. Il peut y avoir dans un texte des références vers un noeud. Emacs gère ces références comme des liens hypertextuels: cliquer sur un lien avec le bouton 2 de la souris suit le lien, et le buffer affiche le texte contenu dans le buffer sur lequel pointe le lien. Je conseille à l'utilisateur intéressé de se repporter au manuel de *Texinfo*, fourni avec toute distribution de *Texinfo* ou à une documentation sur Texinfo se trouvant sur le site d'April (`http://www.april.org`).

Commandes de navigation à travers le manuel *Info*:

- `n`: se déplace sur le noeud suivant
- `p`: se déplace sur le noeud précédent
- `u`: se déplace sur le noeud "père"
- `d`: se déplace sur la racine de l'arbre *Info*
- `f` ou `RET` ou `SPC`: suit une référence à un noeud
- `l`: se déplace sur le dernier noeud visité
- `q`: quitte le mode *Info* et resélectionne le buffer précédemment sélectionné
- `g` *noeud*: se déplace sur le noeud *noeud*
- `s` *regexp*: cherche l'expression régulière *regexp* dans le fichier *Info* sélectionnée
- `TAB`: déplace le curseur sur la prochaine référence hypertextuelle du fichier *Info*
- `M-TAB`: déplace le curseur sur la précédente référence hypertextuelle du fichier *Info*

8.2 Aide contextuelle

L'aide contextuelle est un ensemble de chaînes de caractères associées à des symboles *Lisp*. Il existe plusieurs façons d'accèder à ces chaînes de documentation:

- `C-h c` *séquence*: donne le nom de la commande rattachée à la séquence de touches *séquence*
- `C-h f` *fonction*: affiche la documentation pour la fonction *fonction*
- `C-h k` *séquence*: donne le nom de la commande rattachée à la séquence de touches *séquence* ainsi que sa documentation
- `C-h m`: affiche de la documentation sur le mode majeur utilisé
- `C-h t`: entre dans le tutorial d'Emacs
- `C-h v` *variable*: affiche la documentation sur la variable *variable*
- `C-h w` *commande*: affiche la séquence de touches permettant d'exécuter la commande *commande*

9 Applications

9.1 RMail

9.1.1 Introduction

RMail est un lecteur de mails sous Emacs, écrit en Emacs-Lisp comme tous les modes. RMail permet toutes les fonctionnalités d'un lecteur de mails classique comme *pine elm* ou *netscape*, mais offre toutes les facilités d'Emacs: édition avec les commandes standard d'Emacs, modes, ...

9.1.2 Mails

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

Chaque mail peut posséder des *labels*:

- **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 standard d'emacs, 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-<`)

9.1.3 Listes de mails

Les mails peuvent être visualisés dans une liste (*summary*). Une liste de mails ressemble beaucoup à une liste *dired*: un ensemble de lignes qu'on peut parcourir en se déplaçant avec les flèches. 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, et de leur appliquer des commandes.

Commandes utilisables dans une liste de mails (aussi utilisables directement dans un mail):

- `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 parmis les messages suivants celui contenant l'expression régulière *regexp*

9.1.4 Commandes d'envoi de mail

- `m`: envoi d'un nouveau mail, saisi dans le mode *rmail*
- `r`: reponse à un mail
- `f`: *forward* d'un mail. Ce mail peut être envoyé tel quel à une autre personne
- `M-m`: tente de ré-envoyer un mail dont l'expédition n'a pas réussi

9.1.5 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 de mails.

- `i` *fichier*: lancer RMail sur ce fichier
- `o` *fichier*: exporte le fichier au format *RMail*
- `C-o` *fichier*: export le fichier au format mail propre au système
- `w`: extrait le corps du message avec comme nom le sujet

9.1.6 Variables de RMail

- `rmail-delete-after-output`: si cette variable est fixée à `t`, alors tout mail exporté dans un fichier est marqué comme à effacer.
- `rmail-file-name`: cette variable est le nom du fichier *RMAIL* à ouvrir
- `rmail-output-file-alist`: cette variable permet de fixer le fichier dans lequel un mail sera exporté par défaut. Elle est sous la forme d'une liste d'élements du format: "`(`*regexp* `.` *nom*". Par exemple:

```
(("From:.*celine@.*" . "~/Mail/Celine")
 ("(From|To):.*@boulot\.com" . "~/Mail/Boulot")
 ("Subject:.Java" . "~/Mail/Java")
 ("[Mm]icrosoft" . "/dev/null"))
```

- `rmail-preserve-inbox`: si cette variable est fixée à `t`, alors les messages contenus dans la *mailbox* sont effacés à chaque fois que RMail est exécuté.

9.2 Gnus

Gnus est un lecteur de news écrit en Emacs-lisp. Il est conçu pour lire les news d'*Usenet*, mais aussi pour en poster. *Gnus* s'exécute par `M-x gnus`.

9.2.1 Introduction

Gnus se comporte d'une manière similaire à *RMail*: 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: sourscription, ...
- le buffer `*Summary*`: il contient le sommaire de tous les articles contenus dans un groupe de news. Il est créé lors de la visite d'un groupe de news.
- le buffer `*Article*`: il affiche un article, et est créé par des commandes *gnus* effectuées dans le buffer `*Summary*`. L'utilisateur peut y exécuter des commandes *gnus*.

9.2.2 Utilisation

Gnus lit au démarrage le fichier `.newsrc` dans le répertoire racine de l'utilisateur qui a exécuté Emacs. Ce fichier permet de savoir à quels groupes a souscrit l'utilisateur, et lesquels il a "*killé*".

La lecture de news se fait en deux étapes:

- la sélection de groupe 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 inscrit dans un groupe d'articles, 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 étés *killés* par l'utilisateur
- `l`: liste tous les groupes auxquels a souscrit l'utilisateur et qui contiennent des articles non encore lus
- `u`: souscrit ou dessouscrit (selon le cas) au 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 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 au prochain article non lu
- `p`: déplace la sélection au précédent article non lu
- `SPC`: scrolle le buffer `*Article*` d'une page vers le bas, et sélectionne l'article suivant si le buffer `*Article*` est à la fin
- `DEL`: scrolle le buffer `*Article*` d'une page vers le haut, et sélectionne l'article précédent si le buffer `*Article*` est 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 ">"
- `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 "> "

9.3 Divers

Emacs possède une grande quantité d'applications diverses, des plus utiles aux plus farfelues.

- `Ange-FTP` permet de faire du *FTP* de manière transparente.
- `Dired` est un éditeur de répertoires qui permet l'utilisateur d'utiliser les commandes d'Emacs ainsi que d'autres propres à *Dired*. Dired peut déplacer, copier, renommer des fichiers, mais aussi créér des liens *symboliques* ou *hard*.
- Le mode `shell` est une émulation de session *shell* dans un buffer d'Emacs. L'utilisateur peut exécuter des commandes shell dans ce buffer tout en conservant les raccourcis Emacs et toute la puissance des expressions régulières
- `Spook` permet d'insérer dans le buffer au niveau du point toute une série de mots-clef, générés à partir d'une base de données. Spook, connu désormais sous le nom de `Bruce` est à l'origine une arme contre le NSA Américain, afin d'empécher la reconnaissance automatiques de mails par l'insertion de mots-clef "sensibles" tels que *Qaddafi*, *nuclear* ou *terrorist*, ...
- Le mode `doctor` d'Emacs est sans doute la solution à tous les problèmes existentiels de l'utilisateur: c'est une simulation de consultation de psychologie Freudienne. Ce mode est en fait le célèbre moteur *Elisa* réécrit en *Emacs-lisp*. L'utilisateur lance doctor par `M-x doctor`, et discute avec le psychologue en ecrivant des phrases en langage naturel (en anglais), et finit son discours par une double ligne.
- `Yow` est un petit programme en Emacs-lisp qui génère une phrase dans la zone d'écho à partir d'une base de données de citations de *zippy*. Ce programme parfaitement inutile s'exécute par `M-x yow`.