Emacs

1 Commandes et clés

Chaque fonctionnalité *Emacs* correspond à une fonction *EmacsLisp*. Une fonction *interactive* (appelable par l'utilisateur avec *M-x*) est une *commande*.

Les commandes les plus courantes sont liées à des *clés* (raccourci).

On peut modifier dynamiquement les liaisons entre les clés et les commandes au moyen des commandes :

global-set-key : association valable dans tous les buffers
global-unset-key
local-set-key : association valable dans des buffers avec le même mode majeur
local-unset-key

Avant de lier une clé, on peut vérifier qu'elle n'est pas déjà utilisée au moyen des commandes :

describe-key-briefly (C-h c)
describe-key (C-h k)

EXERCICE 1 – Associez une clé non utilisée (par exemple *C-c c*) à une de vos commandes favorites (par exemple *compile*) n'ayant pas de clé associée.

Cette liaison disparait avec la fin de la session *Emacs*. Pour qu'elle existe dans toutes les sessions, on peut rajouter l'expression suivante dans son fichier *.emacs*.

```
(global-set-key "\C-c c" 'compile)
```

Testez cette solution.

2 Macros Clavier

2.1 Définition d'une macro-clavier

EXERCICE 2 – Récupérer le fichier *fichiers-06.tar.gz*, puis désarchiver le afin d'obtenir le fichier *more.man*, puis placez le dans votre répertoire *~/tmp*. Vérifiez qu'il n'y a pas dans votre répertoire d'accueil un fichier *.emacs* dont vous ignorez le contenu. Si c'est le cas, renommez-le en *.emacs.svg*. Lancez *emacs* et chargez le fichier *more.man*.

La séquence de clés *M-f " M-b "* met entre guillemets (anglais et doubles) le mot désigné par le curseur. Testez cette séquence. Mémorisez cette suite de clés au moyen de la séquence *C-x (M-f " M-b " C-x)*. Vous venez de définir une macro. Remarquez les messages *Defining kbd macro ...* et *Keyboard macro defined*. Placez le curseur sur un mot et faites exécuter la macro au moyen de la clé *C-x e*.

EXERCICE 3 – Nommez cette macro (en utilisant la commande *kmacro-name-last-macro* [pensez à la complétion] ou la clé *C-x C-k n* :

M-x kmacro-name-last-macro RETURN
Name for last kbd macro: *insert-quote* RETURN
M-x insert-quote

Utilisez la commande *insert-quote* [pensez à la complétion].

EXERCICE 4 – Trouver une clé inutilisée parmi les clés *M-"*, *C-"*, *C-x "* et *C-c "* et la lier à la macro *insert-quote* en utilisant la commande *kmacro-bind-to-key*.

Utilisez cette nouvelle clé.

2.2 Sauvegarde de macros pour les prochaines sessions

Remarque – *La commande définie jusqu'à maintenant n'est pas sauvegardée pour les prochaines sessions. Pour préserver la définition de la dernière macro il faut sauvegarder la définition dans un fichier. La définition est en Lisp. Pour sauvegarder la définition d'une macro il faut ouvrir un fichier dans lequel stocker la définition, positionner le curseur à l'endroit où la définition doit être insérée et exécuter la commande* ***insert-kbd-macro****.*

EXERCICE 5 – Nous supposons ici que les macros sont stockées dans le fichier *~/tmp/macros.el*.

— Ouvrez le fichier *~/tmp/macros.el* dans une autre fenêtre :

C-x 4 f ~/tmp/macros.el RETURN

— Exécutez `insert-kbd-macro` pour mettre la définition de la macro `insert-quote` dedans :

C-u M-x insert-kbd-macro RETURN
insert-quote RETURN

— Sauvegardez le fichier ∼*/tmp/macros.el* avec : *C-x C-s*

Remarque : sans l'argument donné par *C-u* la clé associée à la macro ne serait pas sauvée. Essayez les deux versions (avec et sans *C-u*).

EXERCICE 6 — Maintenant nous pourrons réutiliser la commande `insert-quote` et la clé qui y est associée dans une autre session *Emacs*.

— Terminez la session d'emacs : *C-x C-c*
— Lancez une nouvelle session emacs.
— Exécutez la commande *load-file* pour charger le fichier ∼*/macros.el* :

M-x load-file RETURN ∼/tmp/macros.el RETURN

— Ouvrir le fichier *more.man* du répertoire ∼*/tmp* et essayer d'exécuter plusieurs fois la commande *insert-quote* en utilisant la clé associée.

3 Mécanismes de recherche

EXERCICE 7 — Positionner le curseur après le - de la ligne

-c Do not scroll. Instead, paint each screen from the top,

et définir une macro-clavier qui la remplace par

-c: Do not scroll. Instead, paint each screen from the top,

en laissant le curseur à la même place. Pour répéter l'opération de façon efficace, on peut se positionner automatiquement sur le prochain -, au moyen d'une commande de recherche de chaînes de caractères :

C-s I-search: - RETURN (RETURN permet de sortir de la recherche) *C-x e*

Le caractère rencontré n'est pas nécessairement le bon. Recommencer la même opération : *C-s* - On itère la recherche en retapant *C-s* jusqu'à être positionné sur le bon tiret.

EXERCICE 8 — Revenir au début du buffer en utilisant la clef *M-<* et rechercher le mot *lines*. Décrire le mécanisme de recherche incrémentale. Itérer la recherche en tapant à nouveau *C-s*. Ajouter le caractère *point* à la chaîne recherchée. Revenir en arrière avec *C-r*. Enchaîner des recherches jusqu'au message

Failing I-search: lines.

Recommencer pour effectuer une recherche en repartant du début :

Overwrapped I-search: lines.

Remarque — *La chaîne cherchée est mémorisée par* emacs. *Pour la réactiver, il suffit d'itérer* *C-s* *ou* *C-r* *deux fois.*

3.1 Recherche d'un motif

On veut écrire une macro-clavier qui modifie les numéros de page ; par exemple le numéro de page *-3-* devra être remplacé par *Page 3*. Pour écrire cette macro, il faut pouvoir chercher une chaîne de la forme *-chiffre-*. On utilise pour cela des constructions appelées **expressions régulières**. Par exemple, l'expression régulière décrivant le motif *-chiffre-* s'écrit *-[0123456789]-*, ou plus simplement *-[0-9]-*.

Les expressions régulières s'utilisent avec la commande *isearch-forward-regexp* liée à la clé *M-C-s*.

EXERCICE 9 — Définir la macro-clavier traitant les numéros de page.

Les expressions régulières traitées par *emacs* sont construites avec les opérateurs suivants :

Opérateur	Construction
.	Remplace un caractère quelconque
^	Début de ligne
$	Fin de ligne
[-]	Alphabet ([a-f], [P-Z], ou encore [0-9])
*	Répétition du caractère ou de l'intervalle précédent
+	Répétition non vide du caractère ou de l'intervalle précédent

EXERCICE 10 — Recherchez s'il existe des lignes se terminant par `a` ou par `n`.

3.2 Utilisation des tables de *tags*

Les *tags* sont des étiquettes associées aux constructions d'un langage de programmation. On peut construire une table de *tags* associée à un ensemble de fichiers sources au moyen de la commande **etags**.[1] En C, les *tags* sont par défaut associés

— aux fonctions ;
— aux définitions de noms de types : **typedef** ;
— aux définitions de types : **struct**, **union** et **enum** ;
— aux macro-définitions : **#define ...** ;
— aux constantes énumérées (définies au moyen d'**enum**) ;
— aux variables globales ;

Les déclarations de fonctions et de variables externes peuvent également être étiquetées.

EXERCICE 11 – Allez dans le répertoire **bcb** et construisez le fichier **TAGS** en lançant la commande **etags** avec en argument tous les fichiers suffixés par **.c** et **.h** ; examinez le contenu du fichier ainsi créé.

Reconstruisez le fichier en produisant également les définitions externes (recherchez l'option dans la documentation de etags).

EXERCICE 12 – On peut sélectionner une table de tags au moyen de la commande **visit-tags-table**. Sélectionnez la table du répertoire **src** (le nom **TAGS** est choisi par défaut). Recherchez au moyen de la clé M-. un *tag* correspondant à la chaîne **memoire_allouer**.

EXERCICE 13 – La table de *tags* peut être utilisée pour spécifier une liste de fichiers à parcourir, soit pour rechercher un motif spécifié par une expression régulière (**tags-search**), soit pour remplacer un motif par une chaîne (**tags-query-replace**). La recherche ou le remplacement en cours peuvent être relancés au moyen de la clé M-,. Essayez par exemple d'itérer une recherche à partir de

M-x tags-search <RET> chaine.*; <RET>

EXERCICE 14 – Remplacer toutes les définitions et utilisations de **memoire_trace** par **memoire_tracer**.

EXERCICE 15 – Défaire ces modifications au moyen de la commande **tags-search** et des clés M-, et C-_.

4 Marques et régions

La *marque* est la mémorisation de la position courante du point.

C-@ ou C-SPC : affectation de la marque
C-x C-x : échange du point et de la marque

EXERCICE 16 – Observez l'effet de la suite de clés :

C-SPC C-v C-v C-x C-x

Certaines commandes positionnent la marque automatiquement : M-<, M->, C-s, C-r, M-C-s ...

EXERCICE 17 – Essayez M-< C-x C-x M-> C-x C-x C-s C-s C-s RETURN C-x C-x.

Emacs mémorise un *anneau* de marques, que l'on peut parcourir en transmettant un argument à la commande **set-mark-command**, aliasé sur la clé C-u C-SPC.

EXERCICE 18 – Enchaînez plusieurs C-u C-SPC.

La **région** est la partie située entre la marque courante et le point. De nombreuses commandes opèrent sur les régions, comme par exemple **kill-region** liée à C-w

EXERCICE 19 – Définissez une région, détruisez-la, et annulez la destruction par **undo**.

5 Le « kill buffer »

EXERCICE 20 – Se placer au début d'une ligne (C-a) et observer l'effet de

C-k C-y C-y C-y
Revenir en début de ligne avec C-a
C-k C-k C-k C-k C-y C-y
Revenir en début de ligne avec C-a
M-d C-y C-y
Revenir en début de ligne avec C-a
C-k C-k C-u 3 M-d C-n C-a C-y C-n C-n C-y

Toutes ces commandes de suppressions sont des *kill* commandes. Un enchaînement de plusieurs *kill* commandes consécutives à pour effet de mémoriser la suite de caractères détruits dans un buffer appelé *kill buffer*.

EXERCICE 21 – Vérifiez que les commandes de type *delete* lorsqu'elles sont utilisées avec un paramètre d'itération sont traitées comme des *kill* commandes. Vérifiez que C-w est une kill commande. Que fait la commande M-w ?

1. Voir la liste des langages reconnus dans la section **Tags** de la documentation Info d'Emacs ou dans le message d'usage de la commande **etags**.

Remarque – *Le* kill buffer *est en fait un anneau de buffers. Essayez* `C-y M-y M-y` *...*

Exercice 22 – Construisez le damier

```
xxxx    xxxx    xxxx    xxxx    xxxx
xxxx    xxxx    xxxx    xxxx    xxxx
    xxxx    xxxx    xxxx    xxxx    xxxx
    xxxx    xxxx    xxxx    xxxx    xxxx
xxxx    xxxx    xxxx    xxxx    xxxx
xxxx    xxxx    xxxx    xxxx    xxxx
    xxxx    xxxx    xxxx    xxxx    xxxx
    xxxx    xxxx    xxxx    xxxx    xxxx
xxxx    xxxx    xxxx    xxxx    xxxx
xxxx    xxxx    xxxx    xxxx    xxxx
    xxxx    xxxx    xxxx    xxxx    xxxx
    xxxx    xxxx    xxxx    xxxx    xxxx
xxxx    xxxx    xxxx    xxxx    xxxx
xxxx    xxxx    xxxx    xxxx    xxxx
    xxxx    xxxx    xxxx    xxxx    xxxx
    xxxx    xxxx    xxxx    xxxx    xxxx
xxxx    xxxx    xxxx    xxxx    xxxx
xxxx    xxxx    xxxx    xxxx    xxxx
    xxxx    xxxx    xxxx    xxxx    xxxx
    xxxx    xxxx    xxxx    xxxx    xxxx
```

en utilisant le moins de commandes possibles. En particulier, on n'utilisera qu'une seule fois le caractère *X* et deux fois SPC.

6 Remplacements

On veut remplacer tous les *X* du damier par des *%*. On peut utiliser pour cela la commande de remplacement *query-replace* liée à *M-%*.

Exercice 23 – Essayez et commentez les commandes :

M-< M-% X X X X RET % % % % RET
SPC SPC n SPC n n SPC !

Exercice 24 – Il est aussi possible d'utiliser les expressions régulières dans les remplacements avec la clé *M-C-%*. Dans le fichier *more.man*, remplacer tous les mots de quatre lettres commençant par la lettre *m* et finissant par la lettre *e* par *cric*.

6.1 Les buffers

Un buffer est créé pour chaque nouveau fichier que l'on édite. De plus certains buffers sont créés automatiquement par *emacs* : **scratch**, **Help**, **Completions**, ... Ces dernier ne sont par défaut associés à aucun fichier : leur contenu est perdu à la fin de la session.

On peut visualiser la liste des buffers définis à un instant donné au moyen de la commande `list-buffers` liée à la clé *C-x C-b*

Rappel – *On peut redéfinir l'association* buffer → fichier *au moyen de la commande* `write-file` *liée à la clef* `C-x C-w`

Exercice 25 – Sauvegarder le contenu du buffer **Buffer List** dans un fichier */tmp/buffers*. Qu'y a-t-il de différent lorsque l'on refait afficher la liste des buffers ?

Important – *Il est utile, au cours d'une session, de supprimer les buffers inutiles, afin de ne pas saturer la mémoire. On utilise pour cela le buffer* `*Buffer List*` *lui même.*

Dans **Buffer List**, les liaisons sont différentes des liaisons par défaut. Elles correspondent au mode *Buffer Menu.*

Exercice 26 – Consulter la description du mode *Buffer Menu* et supprimer les buffers inutiles. Utiliser le menu des buffers pour sélectionner un buffer.

Remarque – *Il existe également des commandes opérant sur les buffers :* `C-x b`, `C-x k`.

7 Configuration du `.emacs`

Le fichier *~/.emacs* est le fichier de configuration lu par *emacs* à chaque démarrage. Ce fichier contient une liste de commandes écrites en langage Lisp qui sont exécutées lors du démarrage. C'est l'équivalent du fichier *~/.bashrc* pour le *bash*.

Si l'on désire exécuter les commandes du buffer courant, il suffit d'utiliser la commande *eval-buffer*. Pour exécuter un fichier sans le charger dans un buffer, on utilise la commande *load-file*.

EXERCICE 27 – Essayez la fonction *list-colors-display*. Tentez alors de modifier les couleurs de votre *emacs* avec les fonctions suivantes : *set-background-color* et *set-foreground-color* (couleur de fond et d'avant-plan), ainsi que *set-cursor-color* (couleur du curseur).

Il est possible de rajouter directement ces commandes dans le fichier *.emacs* en rajoutant des lignes de la forme ci-après. Le parenthésage correspond à une expression *Lisp*, le premier élément de la parenthèse étant le nom de la fonction invoquée et les éléments suivants ses arguments :

```
(set-cursor-color "MediumBlue")
```

EXERCICE 28 – Rajoutez une clé directement dans le fichier *.emacs*, qui associe à la touche F1 la commande *comment-region*. La syntaxe de la commande est la suivante :

```
(global-set-key [end] 'end-of-line)
```

Le nom de la clé passé en premier paramètre correspond au nom obtenu avec la commande *C-h k*. Le caractère « ' » permet de protéger le nom de la fonction passé en second paramètre.

EXERCICE 29 – La fonction *message* permet d'afficher un message dans la fenêtre de communication d'Emacs. Utiliser cette fonction à la fin du fichier de configuration pour annoncer qu'il a été lu sans erreur en entier.

8 Interaction avec Unix

8.1 Interaction avec le shell

Essayer *M-x shell*.

8.2 Listage et édition de répertoire

8.2.1 Listage

1) listage simplifié : *C-x C-d*
2) listage complet : *C-u C-x C-d*

Attention, *C-x 1* pour retrouver l'affichage du fichier courant en plein écran. Possibilité de substitutions *shell* de noms de fichiers.

8.2.2 Dired

1) passage en mode *édition de répertoire* : *C-x d*, ou visite d'une répertoire
2) descente : *SPC* ou *C-n*
3) remontée : *C-p*
4) marquages et traitements des marques : *d*, *u*, *' '*, *#* et *x*, *~* et *x*
5) copies et renommages : *C R*
6) visite d'un fichier ou d'un répertoire : *f* ou *RET*

Remarque : description du mode *dired* en tapant *C-h m* directement sous *dired*

8.3 Recherche de motifs par grep

1) activation d'une recherche : *M-x grep RET* <*motif*> <*fichier*> ... avec possible substitution *shell* des noms de fichiers
2) sélection automatique du fichier et positionnement du motif : *C-x `*