Emacs (2)

1 Macros Clavier

1.1 Définition d'une macro-clavier

Récupérer le fichier `more.man` dans le répertoire `~renault`, le copier dans son répertoire `~/tmp` puis le charger sous Emacs.

1.1 Exercice Justifier à droite la 12ème et la 15ème ligne de texte, en enchaînant les `M-b SPC`

Puis, mémoriser cette suite de clés en tapant `C-x ( M-b SPC C-x )`

Remarquer les messages `Defining kbd macro...` et `Keyboard macro defined`. Placer le point à la fin d'une ligne et faire exécuter la macro au moyen de la clé `C-x e`. Évaluer le nombre de blancs manquant, et invoquer la macro avec un paramètre de répétition.

1.2 Exercice Nommer la macro insérant un espace :

`M-x name-last-kbd-macro`
Name for last kbd macro : `insere-blanc` Return
`M-x insere-blanc`

Utiliser le mécanisme de complétion de commandes pour taper `name-last-kbd-macro`, puis pour exécuter `insere-blanc` en essayant d'utiliser le mécanisme de complétion.

1.2 Définition de liaisons

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

`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`

1.3 Exercice Trouver une clé inutilisée parmi les clés `M-SPC`, `C-x SPC` et `C-c SPC` et la lier localement à la commande `insere-blanc`.

Utiliser cette nouvelle clé.

1.3 Sauvegarde de macros pour les prochaines sessions

1.4 Exercice

- Définir une macro-clavier plaçant des guillemets ‘‘ et ’’ autour du mot sur lequel est positionné le point.
- Donner le nom `entre-guillemets` à cette macro :

M-x name-last-kbd-macro RETURN entre-guillemets RETURN

- Vérifier que la clé C-x " (ou M-") est libre.
- et la lier globalement à la commande entre-guillemets .

REMARQUE 3 Les commandes définies jusqu'à maintenant ne sont pas sauvegardées 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, mais vous n'avez pas besoin de comprendre Lisp.

Pour sauvegarder la définition d'une macro il faut ouvrir un fichier dans lequel stocker la définition, positioner le curseur à l'endroit où la définition doit être insérée et exécuter la commande insert-kbd-macro.

1.5 EXERCICE Nous supposons ici que vous stockez vos macros dans le fichier ~/tmp/mon_emacs.

- Ouvrir le fichier ~/tmp/mon_emacs dans une autre fenêtre :

  C-x 4 f ~/tmp/mon_emacs RETURN
- Exécuter insert-kbd-macro pour mettre mettre la définition de la macro entre-guillemets dedans :

  C-u M-x insert-kbd-macro RETURN entre-guillemets RETURN
- Sauvegarder le fichier ~/tmp/mon_emacs avec : C-x C-s

1.6 EXERCICE Maintenant nous pouvons réutiliser la commande entre-guillemets avec la clé qui y est associée.

- Terminer la session d'emacs : C-x C-c
- Lancer une nouvelle session emacs.
- Exécuter la commande load-file pour charger le fichier ~/mon_emacs :

  M-x load-file RETURN /tmp/mon_emacs RETURN
- Ouvrir le fichier more.man du répertoire ~/tmp et essayer d'exécuter plusiers fois la commande entre-guillemets en utilisant la clé C-x ".

2 Mécanismes de recherche

2.1 EXERCICE 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.

2.2 EXERCICE Revenir au début du buffer : 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.`

2.3 Exercice Redéfinir la macro précédente en intégrant la recherche. Définir une macro qui remplace `MORE(1)` par `1 : MORE`.

Remarque 4 La chaîne cherchée est mémorisée par *emacs*. Pour la réactiver, il suffit d'itérer `C-s` ou `C-r` deux fois. Enfin, `C-h` permet de remonter d'un cran dans l'historique de la recherche.

2.4 Exercice Observer ce comportement.

2.1 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`.

2.5 Exercice 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

2.6 Exercice Rechercher s'il existe des lignes se terminant par `a` ou par `n`.

3 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

3.1 Exercice Observer 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`...

3.2 Exercice Essayer `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 `set-mark-command`, aliasé sur la clé `C-u C-SPC`.

3.3 Exercice Enchaîner 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`

3.4 Exercice Définir une région, la détruire, et annuler la destruction par `undo`.

4 Le "kill buffer"

4.1 Exercice 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*.

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

Remarque 5 Le *kill buffer* est en fait une anneau de buffers. Essayer `C-y M-y M-y`...

4.3 Exercice Construire 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`.

5 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-%`.

5.1 Exercice Essayer et commenter les commandes :

`M-< M-% X X X X` Return `SPC SPC SPC SPC` Return
`SPC SPC n SPC n n SPC !`

5.2 Exercice Il est aussi possible d'utiliser les expressions régulières dans les remplacements avec la clé `M-C-\%`. Dans le fichier `more.man`, mettez une majuscule à tous les mots de quatre lettres commençant par la lettre `m` et finissant par la lettre `e` par `cric`.

6 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`. Sinon, pour tester son fichier de configuration, il suffit de quitter `emacs` et de le relancer.

6.1 Exercice 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 parenthèsage 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")
```

6.2 Exercice Rajoutez une clé directement dans votre 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.

6.3 Exercice La fonction `message` permet d'afficher un message dans la fenêtre de communication d'Emacs. Utilisez cette fonction à la fin de votre fichier de configuration pour annoncer qu'il a été lu sans erreurs en entier.