

oprintanswers

# Programmation en emacs-lisp

## 1 Organisation de l'environnement emacs

EXERCICE 1 – Si ce n'est pas déjà fait vous devez commencer par organiser votre environnement *emacs* de la manière suivante :

- créez un répertoire *environnement/emacs*
- sauvegardez le fichier d'initialisation *.emacs* original (celui avec lequel votre compte a été initialisé), par exemple en *environnement/emacs/.emacs-original.el*.
- créez un nouveau fichier d'initialisation *.emacs* (à la racine de votre hiérarchie) contenant les lignes :
 

```
(setq load-path
      (cons (expand-file-name "~/environnement/emacs/") load-path))

(load-library "emacs")
```

EXERCICE 2 – Initialisez un fichier *environnement/emacs/emacs.el* minimal contenant les lignes :

```
(tool-bar-mode 0)
(mouse-wheel-mode 1)
(global-font-lock-mode 1 0)
```

EXERCICE 3 – Pour bien comprendre le fonctionnement de ces mécanismes, lancez *emacs* sans chargement de fichier d'initialisation (avec l'option *-q*) et chargez les deux fichiers que vous venez de créer dans deux fenêtres différentes.

EXERCICE 4 – Placez-vous dans la fenêtre *emacs.el* à la fin de l'expression *(tool-bar-mode 0)* et faites évaluer cette expression (clé *C-x C-e*).<sup>1</sup> Faites de même pour les deux autres expressions et observez le résultat.

EXERCICE 5 – Relancez *emacs* normalement (sans l'option *-q*) et vérifiez que l'initialisation se passe correctement.

**Remarque.** Si vous aviez déjà personnalisé votre environnement *emacs* en modifiant les fichiers d'initialisation, vous pourrez répercuter ces modifications sur votre nouveau fichier d'initialisation *emacs.el*. Voici quelques suggestions que vous pourrez explorer après la fin du TD :

- *(standard-display-european t)* : affichage des caractères ISO-LATIN
- *(load-library "iso-insert")* permet de saisir tous les caractères ISO-LATIN sur un clavier *qwerty*, par exemple en tapant 'e pour saisir le caractère é (activation/désactivation au moyen de la commande *iso-accent-mode*)
- *(display-time)*
- *(setq visible-bell t)*

## 2 Écriture d'une fonction interactive

EXERCICE 6 – On rappelle qu'une commande Emacs est une fonction Lisp déclarée *interactive*. Testez ce mécanisme en écrivant une fonction Lisp *(f x)* qui affiche le paramètre quelle reçoit au moyen de la fonction *print*. Pour cela, placez-vous dans le buffer *\*scratch\** (en utilisant la clé *C-x b*, saisissez le code de la fonction, évaluez-la au moyen de la clé *C-x C-e*, et testez-la par *M-x f*. Essayez plusieurs types de paramètres : *n*, *f*, *s*, *p*, etc. (utilisez la documentation en ligne de la fonction *interactive*). Dans le cas où la fonction interactive produit une lecture dans le *minibuffer*, n'oubliez pas de définir un message pour guider la saisie.

EXERCICE 7 – Placez-vous dans le fichier *emacs.el* et écrivez sur le même principe la fonction interactive *confirm-exit* qui reçoit en paramètre une chaîne de caractères et dans le cas où cette chaîne est égale à "yes", provoque la terminaison d'Emacs. Conseils :

- recherchez dans la documentation INFO d'Emacs Lisp la fonction permettant de comparer deux chaînes de caractères.

<sup>1</sup>Cela désactive la barre d'outils qui occupe inutilement de la place.

- Recherchez à quelle fonction est liée la clé `C-x C-c`  
Testez cette fonction et lorsqu'elle est correcte, liez-la à la clé `C-x C-c` dans la table globale.

### 3 Définition d'un *crochet*

EXERCICE 8 – Sous Emacs, un *crochet* est une variable référençant une fonction *accrochée* à un mode. À chaque mode `xxx-mode` est associé un crochet `xxx-mode-hook`. En utilisant ce mécanisme, associez au mode `text` une fonction `text-mode-hook-fonction` qui active le mode `auto-fill` (fonction `turn-on-auto-fill`).

EXERCICE 9 – Vérifiez le fonctionnement de ce mécanisme en chargeant un fichier `test.txt` et en tapant du texte sans et avec le crochet.

### 4 Écriture d'une fonction d'initialisation de fichier en-tête C

Nous disposons maintenant de tous les outils pour ajouter des fonctionnalités au mode C. Nous allons définir une commande permettant d'initialiser un fichier en-tête `nom.h` en plaçant automatiquement au début de ce fichier les directives

```
#ifndef NOM_H
#define NOM_H
```

et à la fin la directive

```
#endif /* NOM_H */
```

EXERCICE 10 – Écrivez une fonction interactive `c-init-header-file` qui extrait le préfixe et le suffixe du nom du fichier associé au buffer courant et termine sur erreur lorsque le suffixe est différent de `.h`. On utilisera les fonctions `buffer-file-name` et `file-name-nondirectory` pour obtenir le nom du fichier, `substring` pour extraire des sous-chaînes d'une chaîne, et `error` pour provoquer une erreur. On initialisera trois variables `name`, `prefix` et `suffix` au moyen d'une construction `let*`.

EXERCICE 11 – Définissez une variable contenant la chaîne `"NOM.h"` où `nom` est le préfixe du nom du fichier et insérer les directives en début et fin du fichier. On pourra utiliser les fonctions `format` et `upcase` pour construire la chaîne `"NOM.h"`, les fonctions `goto-char`, `point-min` et `point-max` pour se déplacer dans le buffer, et `insert-string` et `format` pour insérer les directives.

EXERCICE 12 – Un premier inconvénient de cette fonction est que si elle est invoquée depuis un fichier `.h` contenant déjà du code, on termine l'exécution en fin de buffer. Une solution propre pour retrouver la position courante est d'enfermer l'évaluation de l'expression évaluée dans la question précédente dans une expression `save-excursion`. Modifiez la fonction de cette manière.

EXERCICE 13 – Un second inconvénient de cette fonction est d'insérer les lignes de directives même dans le cas où elles sont déjà présentes. Rajoutez un test qui provoque une terminaison sur erreur lorsque la première ligne est déjà de la forme `#ifndef NOM_H`. On peut extraire une chaîne du buffer au moyen de la fonction `buffer-substring`. Une solution simple mais imparfaite est de tester simplement la présence de la chaîne `"#ifndef "` en début de fichier.

EXERCICE 14 – Pour disposer automatiquement de cette fonction, il suffit maintenant de la placer dans un fichier destiné à recevoir son paramétrage personnel du mode C (par exemple `c-mode.el`) est de provoquer le chargement de ce fichier au moyen du crochet `c-mode-hook`.