Programmation impérative en C

Valisp 5. Histoire de LISP

Olivier Baldellon

Courriel : prénom.nom@univ-cotedazur.fr

Page professionnelle : https://upinfo.univ-cotedazur.fr/~obaldellon/

Licence 2 — Faculté des sciences et ingénierie de Nice — Université Côte d'Azur

Sommaire

:)

Valisp

```
(defun APPLY (FONCTION PARAM ENV)
  (cond ((EQ FONCTION (QUOTE CONSP))   (CONSP (EVAL (CAR PARAM) ENV)))
        ((EQ FONCTION (QUOTE CAR))     (CAR   (EVAL (CAR PARAM) ENV)))
        ((EQ FONCTION (QUOTE CDR))     (CDR   (EVAL (CAR PARAM) ENV)))
        ((EQ FONCTION (QUOTE EQ))      (=     (EVAL (CAR PARAM) ENV) (EVAL (CAR (CDR PARAM)) ENV)))
        ((EQ FONCTION (QUOTE CONS))    (CONS  (EVAL (CAR PARAM) ENV) (EVAL (CAR (CDR PARAM)) ENV)))
        ((EQ FONCTION (QUOTE QUOTE))   (CAR PARAM))
        ((EQ FONCTION (QUOTE COND))    (EVAL-COND PARAM ENV))
        ((CONSP FONCTION) ;; (= (CAR FONCTION) 'LAMBDA)
         (EVAL (CAR (CDR (CDR FONCTION)))
               (MAKE-ENV (CAR (CDR FONCTION))
                         PARAM
                         ENV)))
        (T (APPLY (EVAL FONCTION ENV) PARAM ENV))))
```

- On conserve un tableau de tout les symboles crées
- La fonction `new_symbol`
  - regarde si le symbole existe déjà
  - Si oui, on récupère le pointeur.
  - sinon, on copie la chaîne du symbole en mémoire
- Permet d'éviter de nombreuses allocations
- Attention au ramasse-miettes !
  - Il faut mettre à jour le tableau après le ramasse-miettes

Sommaire

- *Strucure and interpretation of computer programm*
- Introduction de la liaison statique (closures)

Valisp

```
(defun plus-n (n)
    (lambda (x) (+ x n)))

(defvar incr (plus-n 1)) ;; (lambda (x) (+ x 1)

(defun suivant (n)
    (incr n))
```

- Que vaut `(suivant 10)`
  - 11 ?
  - 20 ?

- En fait `incr` vaut `(lambda (x) (+ x n))`
  - Et que vaut `n` ?

- On rajoute une forme spéciale
  - et un nouveau type `closure` (en plus des entiers, symboles, etc.)
  - codé avec un COUPLE (comme les CONS)
  - `(env-de-definition . fonction-lambda)`

Valisp

```
(defun plus-n (n)
    (closure (x) (+ x n)))

(defvar incr (plus-n 1)); (lambda (x) (+ x n))
                        ; dans l'environnement (n . 1) → env
(defun suivant (n)
    (incr n))
```

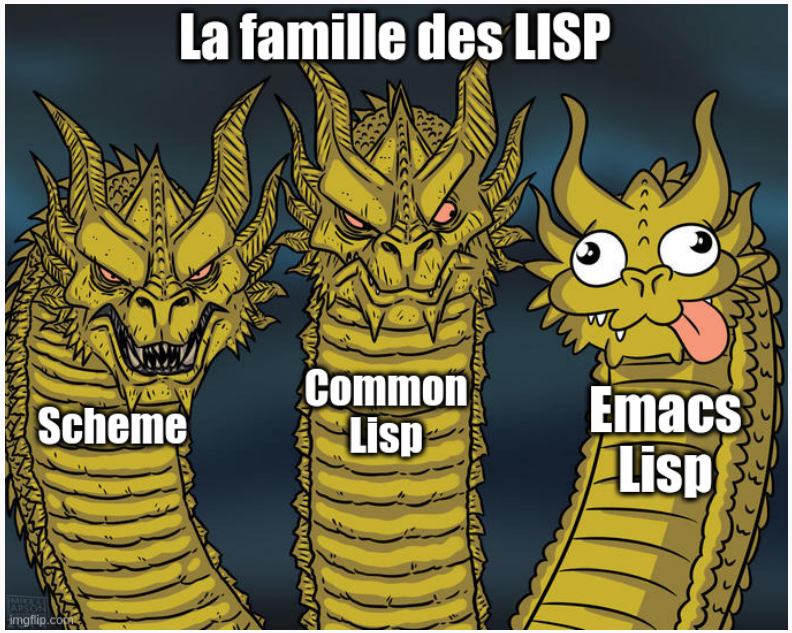
- `sexpr new_closure(sexpr env, sexpr fonction)`
  - attention à l'affichage (boucle infinie)
  - l'environnement local pointant vers l'environnement globale
- `sexpr get_closure_env(sexpr closure)`
- `sexpr get_closure_lambda(sexpr closure)`
- On modifie `apply` pour évaluer le lambda dans l'env. de définition.

Sommaire

:)

:)

Sommaire



- Emacs Lisp
  - C'est le plus récent (1984)
  - mais aussi le plus archaïque
  - et un des plus minimaliste
- Proche de Common Lisp
  - La syntaxe est quasi identiquqe

- Emacs n'est pas un éditeur de texte
- C'est un intepréteur Lisp
  - Chaque touche appelle une fonction
  - On peut combiner ces fonctions pour en écrire de nouvelle
  - Et les associer à des touches
- C'est le dernier héritage des hackeurs du MIT
  - un logiciel avec un contrôle complet
  - un bac à sable
- Aujourd'hui la mode est de tout faire en javascript dans le navigateur
  - Même concept
  - La liberté en moins

Sommaire

Conclusion

- L'informatique est riche en langage divers
  - Lisp, Erlang, Prolog, Smalltalk
  - Ces langages donnent une autre définition au mot *programmer*
- Le C permet de comprendre comment fonctionne les machines
  - Je vous encourage à jeter un coup d'œil à Rust
- Vous devez maîtriser vos outils !
  - Privilégier les outils libres et hackables

Merci pour votre attention
Questions

?

Cours 5 — Histoire de LISP

Licence