École Normale Supérieure — Systèmes et réseaux
Année 2017-2018

TP4 mini-Shell [1]

Le but de ce TP est de programmer un petit interpréteur Shell en OCaml. Pour cela, vous pourrez partir du squelette fourni à l'adresse suivante :

`http://www.di.ens.fr/~pouzet/cours/systeme/tp04/tp-04.tgz`

Cette archive contient les fichiers suivants :

- `msh.ml` : le fichier principal contenant la boucle d'interaction ;
- `msh_ast.ml` : la définition de l'arbre de syntaxe abstraite représentant un script ;
- `msh_evaluator.ml` : les fonctions d'exécution d'un script ;
- `msh_lexer.mll` : le lexeur ;
- `msh_misc.ml` : définition de variables globales ;
- `msh_parser.mly` : le parseur ;
- `msh_printer.ml` : fonctions d'affichage pour le débogage.

Le langage considéré ici est un sous-ensemble de Shell. L'arbre de syntaxe abstraite présente ce sous-ensemble :

```
type command =
  | Nothing
  | Simple_command of simple_command
  | If of command * command * command
  | Connection of command * connector * command

and simple_command =
    { cmd_line: string list;
      cmd_redirects: redirect list;
      cmd_bg: bool; }

and redirect =
  | Input of string          (** < file  *)
  | Output of string         (** > file  *)
  | Output_append of string  (** >> file *)

and connector =
  | Seq     (** ;  *)
  | And     (** && *)
  | Or      (** || *)
  | Pipe    (** |  *)
```

Le constructeur `Nothing` représente une commande qui ne fait rien et termine avec le code de retour `0`. Cette commande est principalement utilisée pour compléter les conditionnelles partiellement définies. Le constructeur `Simple_commande` correspond à une commande simple constituée d'une liste `cmd_line` représentant un nom d'une commande à exécuter avec ses arguments, d'une liste `cmd_redirects` qui sont les redirections des entrées et sorties standard à effectuer et d'un booléen `cmd_bg` indiquant si la commande doit être exécutée en tâche de fond. Le constructeur `If` représente les conditionnelles et `Connection` la composition de commandes.

1. Ces exercices ont été créés par Louis Mandel.

1 Exécution de commandes

Question 1. Complétez le code de la fonction `exec_simple_cmd` pour traiter l'exécution de commandes simples. Cette fonction doit lancer l'exécution d'une commande, attendre sa terminaison et retourner son code de terminaison. Pour le moment, vous ne traiterez pas les redirections ni l'exécution en tâche de fond. (Unix.fork, Unix.exec, Unix.wait)

Question 2. Complétez le code de la fonction `eval_cmd` pour traiter les cas de la conditionnelle et la composition en séquence et par les opérateurs `&&` et `||`.

Question 3. Que se passe-t-il lorsque vous tapez la séquence de commandes suivantes :

```
--> cd /tmp
--> pwd
```

Question 4. Modifier la fonction `exec_simple_cmd` pour traiter la commande `cd` de façon *builtin*. (Unix.chdir, Unix.getcwd)

Question 5. Ajoutez le traitement des redirections.
(Unix.openfile, Unix.dup, Unix.dup2, Unix.stdin, Unix.stdout)

2 Les expansions

Question 6. Que se passe-t-il lorsque vous tapez la commande suivante :

```
--> ls ~
```

Question 7. Traiter le cas où l'un des éléments d'une ligne le commande est la chaîne de caractères `"~"`. (Unix.getenv)

Question 8. Traiter le cas où l'un des éléments d'une ligne le commande est la chaîne de caractères `"*"`. (Unix.opendir, Unix.readdir, Unix.closedir / Sys.readdir)

3 Gestion des signaux

Question 9. Modifiez votre shell pour que, si une commande est en cours d'exécution, un appui sur `Ctrl-C` interrompe la commande et redonne la main à l'utilisateur sans toutefois quitter le shell. (Vous pouvez le tester avec la commande `sleep`.) (Sys.set_signal)

4 Exécution parallèle

Question 10. Ajoutez le traitement des tubes anonymes. (dup2, pipe)

Question 11. Modifiez votre shell pour traiter les commandes lancées en tâche de fond (on ne veut pas de processus zombies).

5 Les variables

Question 12. Ajoutez le traitement de la variable `$?`.

Question 13. Traitez l'expansion des variables globales et la commande `export`.