Entrées-sorties en OCAML

Les fonctions d'entrées sorties calculent une valeur (parfois de type `unit`) et modifient l'état des périphériques d'entrées-sorties :

— modification du buffer du clavier,
— affichage à l'écran,
— écriture dans un fichier
— ou modification du pointeur de lecture.

Deux types prédéfinis `in_channel` et `out_channel` décrivent les canaux de communication d'entrée et de sortie.

Ouverture en lecture Dans un fichier `essai.tex` du répertoire courant, écrivons trois lignes :

```
un
deux
trois et quatre
```

sans retour chariot après « quatre ».

La fonction `open_in` de type `string -> in_channel` permet d'ouvrir un fichier en lecture à partir de son chemin d'accès (ou son nom si le fichier à ouvrir est dans le répertoire courant). Elle ouvre un canal de communication (un flot) avec le fichier s'il existe et déclenche une exception `Sys_error` sinon (notamment si le fichier n'existe pas).

Notre fichier est ouvert avec l'instruction

```
let ic = open_in "essai.txt";; (*création d'un canal de com. vers essais.txt*)
```

On accède aux lignes du fichier grâce à la fonction

```
 # input_line;;
- : in_channel -> string = <fun>
```

Ci-dessous, nous lisons et affichons la première ligne du fichier :

```
let s = input_line ic in Printf.printf "%s\n" s;;
un
- : unit = ()
```

Lisons donc les deux autres lignes :

```
# let s = input_line ic in Printf.printf "%s\n" s;;
deux
- : unit = ()
# let s = input_line ic in Printf.printf "%s\n" s;;
trois et quatre
- : unit = ()
```

Si on essaye de lire une nouvelle ligne, on se doute bien qu'on va au devant d'un problème : une exception `End_of_file` sera soulevée. Ne tentons pas le diable et fermons le canal de communication `ic` :

```
# close_in ic;;
- : unit = ()
```

Le canal `ic` est fermé. Toute tentative de lire une ligne de `essai.txt` se solde par une exception `Sys_error "Bad file descriptor"`.

En résumé, pour lire et afficher toutes les lignes d'un fichier et le fermer proprement, il suffit de rentrer dans une boucle infinie qui affiche les lignes une à une et de récupérer l'exception `End_of_file` qui finira par arriver. On ferme alors le canal.

```
 let ic = open_in "test1.tex" in
let rec lire () =
  let s = input_line ic in Printf.printf "%s\n" s;
  lire();
in
try
  lire();
with End_of_file ->  close_in ic;;
```

Après compilation et exécution, le contenu du fichier `test1.tex` s'affiche (s'il existe).

Ouverture en écriture Quand on ouvre un canal de communication en écriture, le fichier correspondant est ouvert s'il existe ou créé s'il n'existe pas.

La fonction `open_out` permet d'ouvrir le fichier en mode écriture « avec écrasement ». La fonction `close_out` referme le canal

```
let oc = open_out "sortie.txt" in close_out oc;;
```

Nous avons juste ouvert puis refermé le fichier mais il a bien été créé :

```
$ ls sort*
sortie.txt
```

La fonction `output_string` permet d'écrire une chaîne de caractère dans le fichier :

```
 # let oc = open_out "sortie.txt" in
output_string oc "un";
output_string oc "deux";
output_string oc "trois";
close_out oc;;
        - : unit = ()
```

Attention, aucun saut de ligne n'a été inséré, il ne faut donc pas oublier les `\n` si on veut passer à la ligne :

```
$ cat sortie.txt
undeuxtrois
```

Plus proche de ce que nous connaissons est la fonction `Printf.fprintf` qui reconnaît les spécifieurs de format :

```
let oc = open_out "sortie.txt" in
Printf.fprintf oc "%f\n" 3.45;
Printf.fprintf oc "%d\n" 26;
Printf.fprintf oc "%s\n" "fini";
close_out oc;;
```

Vérification

```
$ cat sortie.txt
3.450000
26
fini
```