TRAVAUX PRATIQUES XIX

Manipulation de fichier

Ce TP est en OCaml. L'objectif est de présenter le b-a-ba de la mnipulation d'un fichier.

A Permissions

Pour lire/écrire/exécuter un fichier, il faut y être autorisé·e ! Dans un terminal, la commande `ls -l` permet de lister les fichiers du répertoire courant et leurs permissions. Les permissions sont une suite de 9 caractères, qui se comprend ainsi :

- Le premier caractère est `d` s'il s'agit d'un dossier, `l` s'il s'agit d'un lien, et `-` sinon.
- Les trois caractères suivants sont `r` (`r` ead) `w` (`w` rite) et `x` (e`x` ecute) et représente les permissions de lire le fichier, modifier le fichier, et exécuter le fichier *pour l'utilisateur courant*. Si une de ces permissions est absente, il y a un `-` à la place.
- Les trois caractères suivants sont la même chose, mais pour le *groupe* de l'utilisateur courant.
- Les trois derniers sont la même chose, mais pour *tout le monde*.

0. Regardez les permissions dans votre dossier personnel. Allez voir celles de `/home` . Commentez.

B Lecture d'un fichier en OCaml

B.1 `open_in`, `input_line` et `close_in`

En OCaml (comme en C et en Python), pour lire un fichier il faut commencer par l'ouvrir. L'ouverture du fichier crée une « variable » qui permet de manipuler le fichier. Une fois que l'on a fini de lire le fichier, il faudra le refermer (ce qui détruira cette « variable »). Cette « variable » est appelée un **descripteur de fichier**, ou encore un **canal**.

Pour ouvrir un fichier nommé `file.ml` en OCaml :

```
let fichier = open_in "file.ml"
(* fichier est le descripteur de fichier obtenu *)
```

On peut alors lire une ligne du fichier avec `input_line` : cette fonction prend en entrée un descripteur de fichier et renvoie la première ligne non-encore lue du fichier (donc de type `string`) :

```
let line = input_line fichier
```

Une fois que l'on a fini de lire ce que l'on voulait lire, on ferme le fichier avec `close_in` :

```
let _ = close_in fichier
```

Remarque. Comme `close_in` renvoie `unit` et que `()` est la une seule valeur de type `unit`, on peut écrire :

```
let () = close_in fichier
```

1. À l'aide de ces trois fonctions, faites un code qui affiche la première ligne de `le_lac_lamartine.txt` . *Mentionnons que* `print_string` *affiche une string, et que* `print_endline` *affiche une string en rajoutant un retour à la ligne à la fin.*

2. Modifiez votre code pour qu'il affiche les 4 premières lignes.

B.2 `try...with` et `input_line`

Lorsque le fichier est entièrement lu, `input_line` délenche une exception (une « erreur ») nommée `End_of_file` .

Nous avons donc envie d'écrire un code qui ressemble à « si `input_line` ne fait pas d'erreur Alors ... Sinon (si elle a levé `End_of_file`) ... ». Autrement dit, on veut faire une disjonction de cas non pas sur la valeur renvoyée par `input_line`, mais sur si la fonction a renvoyé quelque chose ou bien déclenché une exception. En OCaml, cela se fait avec la construction `try ... with` :

Pour l'illustrer, voici une fonction qui renvoie soit la prochaine ligne de fichier, soit "FINI" si il n'y a plus de ligne à lire :

```
let demo fichier =
  try
    input_line fichier
  with
    | End_of_file -> "FINI"
```

Notez que la syntaxe du `with` ressemble à un filtrage : c'est voulu, on peut faire différent cas dans le `with` pour distinguer différentes exceptions[1].

Remarque. Pour un exposé un peu plus exhaustif des exceptions, je vous renvoie à la partie correspondante du cours sur les bonnes pratiques.

3. Écrire une fonction récursive qui affiche tout un fichier. Pour cela, elle récupère une ligne, l'affiche puis s'appelle récursivement. Si la récupération d'une ligne lève une exception, elle s'arrête (et renvoie `()`).
4. Testez avec Le Lac !

C Écriture dans un fichier

L'écriture dans un fichier fonctionne de manière symétrique :

- `open_out` prend en argument le nom d'un fichier et l'ouvre (ou le crée si besoin) pour écrire dedans. Le « curseur » est placé à la fin du fichier.
- `output_string` prend deux arguments (un descripteur de fichier et une string) et écrit la string à la fin du fichier.
- `close_out` ferme le fichier.

5. À l'aide des fonctions de lecture et d'écriture, écrivez un code qui recopie le contenu de `le_lac_lamartine.txt` dans un nouveau fichier `copie.txt` .

D Manipulation des lignes

Dans la vraie vie, on veut rarement uniquement récupérer une ligne : on veut les manipuler.

D.1 Découpe de string

Une première manipulation courante consiste à séparer la ligne en ses mots, c'est à dire à « découper sur les espaces ». La fonction `String.split_on_char : char -> string -> string list` permet cela : elle prend en argument un caractère et une string, et renvoie la liste des string successives obtenues en découpant l'entrée sur le caractère indiqué.

Exemple. `String.split_on_char ' ' "bonjour le monde"` vaut `["bonjour"; "le"; "monde"]` .

La fonction « réciproque » de `String.split_on_char` est `String.concat : string -> string list -> string` . Elle prend en argument un séparateur (la première string), une liste de string, et colle les éléments de la liste de string en insérant entre eux le sépareteur.

Exemple. `String.concat " " ["bonjour"; "le"; "monde"]` vaut `"bonjour le monde"` .

6. Sachant que la fonction `List.sort String.compare` permet de trier une liste de string, créez un fichier `lac_trie.txt` qui contient les mêmes lignes que Le Lac, mais où chacune d'entre elles est triée.
   La première ligne du résultat final devrait être `Ainsi, de nouveaux poussés rivages, toujours vers`

1. C'est comme une alarme : il y a différents signaux d'alarmes qui ont différents sens, et on ne veut donc pas réagir de la même façon aux différents signaux.

D.2 Conversion vers un autre type

La fonction `input_line` n'est pas magicienne : elle ne peut pas deviner le « bon type » des données à lire, et elle renverra toujours une ligne comme une string. Pourtant, ce n'est parfois pas le type adapté : regardez par exemple `entiers.txt`. On aimerait bien lire sa ligne comme une *liste d'entiers*.

On va réussir à le faire en combinant les fonctions suivantes :

- `String.split_on_char`
- `int_of_string` qui convertit une string en l'entier qu'elle contient (si la string ne correspond pas à un entier, elle lève l'exception `Failure "int_of_string"`)
- `List.map` : cf la documentation du module `List`, çad `https://ocaml.org/manual/5.3/api/List.html`.

7. Écrivez un programme qui calcule et affiche la somme des entiers de `entiers.txt`, en le lisant.

8. Résolvez le jour 7 de l'advent of code 2024 : `https://adventofcode.com/2024/day/7`. Il vous faudra lire l'entrée dans un fichier (pas évident[2]), puis résoudre le problème donné (c'est un retour sur trace pas trop méchant).

2. Vous pouvez le faire avec uniquement ce que je vous ai donné dans ce TP ; ou aller chercher d'autres fonctions utiles dans le module `String`