TP de programmation n° 5

Julien Reichert

Mardi 12 novembre 2013

Les entrées et sorties

Pour déboguer vos programmes, il vous sera très utile de faire des tests qui consistent en règle générale à insérer une ligne à un endroit particulier de votre code. Cette ligne, dont le contenu dépend du style du programmeur, se résume à `print_endline "Coucou";`.

Dans ce TP, nous allons un peu plus loin dans ce qui concerne le module `Printf` et les entrées/sorties, afin de rendre les programmes interactifs.

1 Fonctions d'impression de base

Pour tous les types standards, il existe une fonction d'impression. On regrette l'absence d'une fonction polymorphe (que l'on compensera par les formats plus bas). Il y a donc `print_char`, `print_string`, `print_int` et `print_float`. Par chance, `print_bool` nous a été épargné. Notez que ces fonctions mettent leur argument dans une sorte de buffer, et n'impriment le résultat qu'après une action entraînant un "flush", par exemple par la fonction `flush_all ()`, par la fonction `print_newline()`, par l'impression d'un retour à la ligne plus généralement, ou par la fin d'un programme.

Ainsi, si vous écrivez successivement un `print_string` et une instruction qui bloque l'exécution (en attendant une saisie ou un certain temps), vous ne verrez pas le résultat tout de suite en l'absence de ce "flush".

Le raccourci `print_endline` combine `print_string` et `print_newline ()`.

Quant à la lecture, elle est faite par `read_line` et ses variantes pour d'autres types `read_int` et `read_float`, qui attendent une saisie de l'utilisateur, validée par un saut à la ligne.

2 Travailler avec des fichiers

OCaml peut interagir avec l'entrée et la sortie standards, mais aussi avec des fichiers, qui font alors office d'entrée ou de sortie suivant qu'ils soient ouverts en lecture ou en écriture.

Lorsqu'on ouvre un fichier, un canal d'entrée ou de sortie est ouvert sur ce fichier et positionné au début. S'il est ouvert en écriture, il est éventuellement écrasé ou créé.

On écrira alors `let sortie = open_out "chemin_vers_le_fichier" in ...` et on terminera de préférence par l'instruction `close_out sortie` car les fichiers ouverts doivent toujours être refermés.

Pour ouvrir en lecture, vous l'aurez deviné, c'est `open_in` et `close_in`.

Maintenant qu'un fichier est ouvert, que peut-on y faire ? Pas imprimer avec les fonctions ci-dessus, réservées au toplevel. On utilise alors, entre autres, les fonctions `output_char`, `output_string` et `output_value`, qui encode dans la sortie (en premier argument comme pour les autres fonctions) la valeur en deuxième argument, de type quelconque.

Remplacer `in` par `out` dans les fonctions du paragraphe précédent donne les fonctions de lecture.

En plus des canaux d'entrée et de sortie, il existe un canal d'erreurs, par défaut `stderr`.

3 Le module d'impression

Répondant au doux nom de `Printf`, ce module de la bibliothèque standard permettra d'utiliser des formats dans vos chaînes de caractères. Qu'est-ce qu'un format ? Il s'agit en quelque sorte de mettre une variable dans une chaîne afin d'éviter une succession de fonctions. Par exemple, pour annoncer un nombre a priori inconnu, au lieu d'écrire

```
print_string "Vous avez "; print_int n; print_endline " nouveau(x)
message(s)";;
```

on se contentera de

```
Printf.printf "Vous avez %d nouveau(x) message(s)\n" n;;
```

Les formats peuvent être aussi nombreux que l'on veut dans la chaîne de caractères, les variables devront alors être ajoutées dans l'ordre après cette chaîne, comme si on avait affaire à une fonction ayant plusieurs arguments.

Les codes principaux sont `%d` pour un entier signé (d pour décimal, dans le sens « base dix »), `%f` pour un flottant, `%s` pour une chaîne de caractères, `%c` pour un caractère...et `%%` pour le symbole `%`.

La fonction `printf` imprime le résultat sur la sortie standard, la fonction `fprintf` l'imprime sur un canal de sortie précisé en premier argument, la fonction `eprintf` l'imprime sur un canal d'erreur précisé en premier argument et la fonction `sprintf` retourne une chaîne de caractères.

Le type de sortie de toutes ces fonctions n'est pas fixe, vu que les arguments manquants ne posent pas de problème en soi. Par exemple, le type de

```
Printf.sprintf "Vous avez %d nouveau(x) message(s)\n";;
```

sera `int` $\rightarrow$ `string`.

4 Exercices

Exercice 1 : Écrire un programme qui fait deviner un nombre entre 0 et un seuil donné, les réponses du programme à une saisie de l'utilisateur sont qu'elle est exacte, supérieure ou inférieure au nombre à deviner. Le nombre d'essais sera annoncé à la fin.

Exercice 2 : Écrire un programme qui imprime une matrice, donnée en argument, avec un peu d'ASCII art (colonnes au moins, si possible lignes). Il faudra se poser la question de la taille de chaque case afin de garder l'alignement, voire l'esthétique.

Exercice 3 : Écrire un programme qui lit un fichier texte composé uniquement de caractères alphanumériques et qui note dans un autre fichier le nombre d'occurrences de chacun de ces 36 caractères.

Exercice 4 : Écrire un protocole cryptographique de votre choix [1] qui chiffre sur place un message dans un fichier texte.

1. pas trop trivial tout de même