

TP* : Entrées-sorties en C et Ocaml

Avant toute chose, récupérer les fichiers à votre nom dans les ressources pédagogiques (répertoire TP_IO).

Cours : rappels succincts de syntaxe

Pour plus amples détails, consulter la documentation des fonctions ci-après.

En C

- Ouverture d'un fichier : `fopen(<chemin fichier>, <mode de lecture>)`. Le premier argument est une chaîne de caractère correspondant au chemin du fichier et le deuxième une chaîne de caractères valant "w" pour ouvrir le fichier en écriture et "r" pour ouvrir le fichier en lecture. Si on ouvre un fichier qui n'existe pas en écriture, il est créé ; s'il existe, son contenu sera écrasé. La valeur de retour de `fopen` est de type `FILE*`.
- Fermeture d'un flux `f` (de type `FILE*`) : `fclose(f)`.
- Ecriture dans un fichier : `fprintf(<flux sur lequel écrire>, <chaîne formatée>, ...)` S'utilise comme `printf` sauf qu'on précise en premier argument où écrire.
- Lecture dans un fichier : `fscanf(<flux sur lequel lire>, <chaîne formatée>, <adresse1>, ..., <adressen>)`. Par exemple, si on sait qu'une ligne contient deux entiers séparés par un espace et que `a` et `b` sont de type `int`, `fscanf(f, "%d %d\n", &a, &b)` permet de les récupérer dans `a` et `b`. La valeur de retour de `fscanf` est un entier correspondant au nombre de paramètres correctement extraits (dans l'exemple si tout se passe bien cette valeur serait 2). Si aucune donnée ne peut être extraite, EOF est renvoyé.

En Ocaml

- Ouverture d'un fichier en lecture : `let f = open_in <chemin fichier>`. Si le fichier n'existe pas, lève l'exception `Sys_error "No such file or directory"`.
- Fermeture de ce flux : `close_in f`.
- Ouverture d'un fichier en écriture : `let f = open_out <chemin_fichier>`. Si le fichier n'existe pas, il est créé. Si le fichier existe déjà, le contenu sera écrasé.
- Fermeture de ce flux : `close_out f`.
- Lecture d'une ligne sur le canal `f` : `input_line f`. La ligne est "consommée" : le prochain appel à `input_line` donnera la chaîne correspondant à la ligne suivante. Si toutes les lignes ont été consommées, lève l'exception `End_of_file`.
- Ecriture de `s` sur un canal de sortie `f` : `output_string s f`. On peut aussi utiliser `Printf.fprintf f "%s" s`.

TP : résoudre les énigmes

Vous avez à votre disposition un fichier `Consignes_chiffre.txt`, pour le moment illisible. Ce document renferme les consignes du TP. Il ne contient que des lettres minuscules non accentuées et des symboles de ponctuation et blancs.

Chaque lettre minuscule de ce document a été chiffrée à l'aide d'un chiffrement de César : on s'est donné un entier d fixe, appelé *décalage* et pour chaque lettre ℓ dans le texte d'origine (dit : texte clair) le traitement suivant a été appliqué :

- Calculer la position $i \in \llbracket 0, 25 \rrbracket$ de ℓ dans l'alphabet.
- Calculer $i' = (i + d) \bmod 26$.
- Déterminer la lettre ℓ' dont la position est i' dans l'alphabet.
- Remplacer la lettre ℓ dans le document d'origine par ℓ' .

Les caractères autres que les lettres minuscules ne sont pas chiffrés. Le décalage de ce chiffrement est donné par le nombre d'espaces dans `Consignes_chiffre.txt`.

L'objectif est de résoudre les différentes énigmes décrites dans les consignes en respectant ces contraintes :

- Le déchiffrement des consignes doit se faire en C.
- La première partie de l'énigme décrite dans les consignes doit se faire en C.
- La deuxième partie de l'énigme décrite dans les consignes doit se faire en Ocaml.