Initiation à la Programmation en C (L1 CPEI)*

TP 8

03/04/2019

§1. Lecture et écriture dans un fichier

Ouverture. Pour interagir avec un fichier, la première chose à faire est d'inclure la *bibliothèque standard* `stdio` du langage C:

```c
#include <stdio.h>
```

Ensuite, nous pouvons ouvrir un fichier, disons `test.txt`, comme suit :

```c
FILE* file = fopen("test.txt", "r");
```

On utilise le mode `"r"` (*read*) pour *lire* et `"w"` (*write*) pour *écrire*.

Attention ! En ouvrant avec le mode `"w"` (*write*), le contenu du fichier est effacé ! Pour écrire à partir de la fin du fichier sans écraser le contenu existant, vous pouvez utiliser le mode `"a"` (*append*).

Attention ! Si l'ouverture du fichier a échouée, le pointeur *file* vaudra `NULL`. Pensez donc à vérifier que l'ouverture a bien eu lieu:

```c
if (!file) { // Ou de manière équivalente if(file == NULL)
    // Erreur !
} else {
    // Le fichier est bien ouvert
}
```

*Cours donné par prof. Roberto Amadio. Moniteur 2019 : Cédric Ho Thanh. TPs/TDs basés sur ceux des précédents moniteurs : Florien Bourse (2017), Antoine Dallon (2018). Autres contributeurs : Juliusz Chroboczek, Gabriel Radanne.

Lecrure et écriture. Pour lire et écrire depuis et vers un fichier on utilise les fonctions `fscanf` et `fprintf`, respectivement, qui sont complètement analogues aux fonctions `scanf` et `printf` que vous connaissez déjà. Par exemple, pour lire un entier:

```
    FILE* file = fopen("test.txt", "r"); % Ouverture en mode lecture
    int x;
    fscanf(file, "%d", &x); // Comme scanf, a l'exception du premier argument
```

De même, pour écrire un entier:

```
    FILE* file = fopen("test.txt", "r"); % Ouverture en mode lecture
    int x = 123;
    fprintf(file, "%d", x); // Comme printf, a l'exception du premier argument
```

Lire une chaine de caractères. Pour lire une ligne d'un fichier, nous devons d'abord déclarer un *buffer* d'une taille donnée.

```
char buffer[2048]; // Un buffer pouvant stocker 2048 caractères
char* resultat = fgets(buffer, sizeof(buffer), file);
if (resultat != NULL) { // Important !
    printf("\%s", buffer);
}
```

Si la fin du fichier a été atteinte, ou si une erreur est survenue, la fonction `fgets` renvoie la valeur `NULL`. Sinon, `buffer` contient une suite de `char` terminée par `"\0"`, comme vu en cours.

Fermeture. Enfin, quand nous avons fini de manipuler le fichier, il est important de le fermer :

```
fclose(file);
```

Attention ! Une fois que le fichier est fermé, il n'est plus possible d'y écrire en utilisant `fprintf` ou de le lire en utilisant `fscanf` ou `fgets`.

Exercice 1. Écrivez un programme qui lit son propre fichier `.c` ligne par ligne (en supposant qu'il est dans le même dossier) et l'affiche sur le terminal.

Exercice 2 (Cryptographie pour débutant)**.** Le ROT13 est un algorithme simpliste de chiffrement qui est parfois utilisé pour cacher certains mots dans les listes de discussion. Encoder une chaîne en ROT13 consiste à remplacer chaque caractère alphabétique (minuscule ou majuscule, sans accent) par le caractère qui est 13 positions plus loin dans l'alphabet, *modulo* 26 ; les autres caractères restant inchangés. Par exemple, la chaîne `"Bonjour."` est codée en `"Obawbhe."`.

(1) Comment fait-on pour décoder une chaîne codée en ROT13 ?

(2) Écrivez une fonction `char* rot13(char* dest, int size, char* src);` qui applique le codage ROT13 à la chaîne `src` et place le résultat dans `dest`, qui peut contenir au plus `size` octets. Elle retournera `dest` en cas de succès, ou `NULL` si `dest` était trop petit. En aucun cas vous ne devrez écrire plus de `size` caractères ! Écrivez une fonction `main` pour tester votre fonction.

(3) Modifiez votre fonction `main` afin que votre programme lise un ficher `fable.txt` (disponible sur Moodle), et le chiffre en utilisant ROT13 dans le fichier `rot13.txt`.

§2. Les arguments en ligne de commande

Dans le terminal, vous utilisez souvent des *arguments en ligne de commande*, par exemple `cp f1.txt f2.txt`. Il est possible de les utiliser dans un programme C en les indiquant dans le main de la manière suivante :

```c
int main (int argc, char* argv[]) {
    // Du code...
    return 0 ;
}
```

La variable `argc` représente le nombre d'arguments, alors que `argv` est un tableau de chaînes de caractères contenant lesdits arguments. Notez que la taille du tableau `argv` est `argc`.

Exercice 3. Écrivez un programme `perroquet` qui recopie tous ses arguments. Par exemple

```
./perroquet Il y a de l echo
```

affichera

```
Il y a de l echo
```

À noter : Ce programme existe déjà sur votre système et s'appelle `echo`.

Exercice 4. En se basant sur le premier exercice, écrivez un programme qui lit le fichier indiqué par le premier argument en ligne de commande, et le recopie dans le deuxième argument.

Exercice 5. Modifiez votre programme ROT13 de l'exercice 2 afin qu'il affiche le codage ROT13 des paramètres de ligne de commande. Par exemple, si vous tapez

```
1 ./rot13 Bonjour, le C.
```

votre programme affichera

```
Obawbhe, yr P.
```