TP 5 – Introduction aux fichiers texte et synthèse sur les chaînes de caractères

1 Objectifs du TP

Ce TP a pour but de découvrir comment :

1. créer un fichier texte ;
2. lire un fichier texte ;
3. écrire un petit programme manipulant des fichiers textes et des chaîne de caractères.

2 Généralités sur les fichiers

Les signatures des sous-programmes de manipulation de fichiers sont dans le fichier `<stdio.h>`.

Pour utiliser un fichier, il faut déclarer une variable pointeur de type `FILE*`. La première opération à faire est l'ouverture du fichier qui permet de lier la variable déclarée dans le programme à un fichier du système de fichier de votre ordinateur.

Cette opération est réalisée par la fonction `fopen` qui a deux arguments de type chaîne et qui retourne un pointeur de type `FILE*`. Le premier argument contient le chemin d'accès au fichier et le second le mode d'ouverture. Nous n'étudierons ici que les modes `"w"`, écriture (write) et `"r"`, lecture (read).

La fonction retourne un pointeur nul si l'opération échoue, par exemple si on ouvre en lecture un fichier inexistant ou si on ouvre en écriture un fichier que l'on a pas le droit d'écrire.

Quand votre programme a fini d'utiliser un fichier, il faut le fermer avec la fonction `fclose` qui a pour argument le pointeur retourné par `fopen`.

Créer un nouveau répertoire pour y mettre les fichiers de ce TP. Dans ce répertoire, créer un fichier *fichiers.c* contenant le code suivant :

```
1.      #include <stdio.h>
2.      void main() {
3.          FILE* fichier;
4.          fichier = fopen("f1.txt", "w");
5.          if (fichier == NULL) {
6.              printf("Erreur à l'ouverture du fichier f1\n");
7.          } else {
8.              printf("Ouverture du fichier f1 réussie\n");
9.              fclose(fichier);
10.         }
11.     }
```

Q. 1 Exécuter le programme puis avec la commande « `ls -l f1.txt` » vérifier que le fichier a été créé et donner sa taille.

Q. 2 Avec un éditeur de texte ou avec la commande « `cat` », taper quelques caractères dans le fichier `f1.txt` et ré-exécuter votre programme. Que constatez-vous ?

Q. 3 Supprimer le droit en écriture sur le fichier `f1.txt` et ré-exécuter le programme. Que se passe-t-il ?

Ajouter le code suivant à la fin de votre programme :

```
1.          fichier = fopen("f2.txt", "r");
2.          if (fichier == NULL) {
3.              printf("Erreur à l'ouverture du fichier f2\n");
4.          } else {
5.              printf("Ouverture du fichier f2 réussie\n");
6.              fclose(fichier);
7.          }
```

Q. 4 Exécuter le programme. Que se passe-t-il pour le fichier f2 ?

Q. 5 Créer le fichier `f2.txt` avec la commande « `touch f2.txt` » et ré-exécuter votre programme. Que se passe-t-il pour le fichier f2 ?

3 Écriture et lecture d'un fichier

Pour écrire dans un fichier texte, le plus simple est d'utiliser la fonction `fprintf`. Elle est similaire à `printf` que vous connaissez déjà. La seule différence est un argument supplémentaire, le premier, qui contient le pointeur désignant le fichier.

Créer un nouveau fichier `fichiersEcriture.c` contenant le code suivant :

```
1.      #include <stdio.h>
2.      #include <string.h>
3.      #include <stdlib.h>
4.      void main() {
5.          FILE* fic;
6.          int lgLigne;
7.          fic = fopen("texte.txt", "w");
8.          if (fic == NULL) {
9.              printf("Erreur à l'ouverture du fichier texte\n");
10.         } else {
11.             char* ligne = NULL;
12.             size_t taille = 0;
13.             printf("Entrer un texte terminé par une ligne vide\n\n");
14.             lgLigne = getline(&ligne, &taille, stdin);
```

```
15.             while (lgLigne > 1) {
16.                 fprintf(fic, "%s", ligne);
17.                 lgLigne = getline(&ligne, &taille, stdin);
18.             }
19.             free(ligne);
20.             fclose(fic);
21.         }
22.     }
```

Q. 6 Exécuter le programme et vérifier que le fichier `texte.txt` contient bien les lignes que vous avez saisies.

Une autre façon d'arrêter la saisie est de simuler une fin de fichier sur l'entrée standard en tapant au terminal le caractère `ctrl-D` en début de ligne. Quand la fonction `getline` est appelée alors que la fin de fichier est atteinte, elle retourne -1.

Q. 7 Modifier votre programme pour que la saisie s'arrête uniquement sur une fin de fichier.

Pour lire un fichier texte, le plus simple est d'utiliser la fonction `getline` que vous connaissez déjà pour l'entrée standard, en remplaçant `stdin` par le pointeur du fichier que l'on veut lire.

Q. 8 En vous inspirant du programme contenu dans `fichiersEcriture.c`, créer un nouveau programme, rangé dans `fichiersLecture.c`, qui fait le traitement inverse, c'est à dire qui affiche à l'écran le contenu du fichier `texte.txt`.

Après qu'une opération de lecture ait rencontré une fin de fichier, une autre façon de tester cette condition est d'utiliser la fonction `feof`. Cette fonction a pour argument un pointeur de fichier. Elle retourne une valeur non nulle si une opération de lecture sur ce fichier a rencontré une fin de fichier.

Q. 9 Modifier votre programme précédent en utilisant la fonction `feof` pour tester la fin du fichier. Vous pouvez remarquer que la variable `lgLigne` ne sert plus. On peut donc la supprimer.

4 Exercice de synthèse sur les chaînes et les fichiers texte

On souhaite gérer un fichier contenant un répertoire clients très simple.

Télécharger l'archive disponible dans le cours et extraire le répertoire Synthèse. Placer vous dans ce répertoire. Vous allez y trouver 4 fichiers : le fichier `répertoire1.txt` contenant un exemple de répertoire client, le fichier `clients.h` contenant les déclarations nécessaires et les signatures des fonctions à développer, le fichier `synthese.c` qui vous permettra de tester les fonctions à développer et le fichier `Makefile`.

⚠ Dans cet exercice, on va considérer que les lignes du répertoire clients sont bien formées, c'est à dire quelles contiennent bien les bonnes informations séparées par le caractère `':'`.

Créer le fichier clients.c qui contiendra le code suivant :

```
1.      #include <stdio.h>
2.      #include <string.h>
3.      #include <stdlib.h>
4.      #include "clients.h"
```

Q. 10 Ajouter à ce fichier le corps de la fonction `charger`. Pour extraire les trois champs prénom, nom et numéro, vous pouvez utiliser la fonction `strchr` pour rechercher les positions des deux séparateurs dans la ligne lue. Tester avec le programme `synthese.c` dans lequel vous aurez commenté les appels aux fonctions non encore développées.

Q. 11 Ajouter à ce fichier le corps de la fonction `ranger`. Tester avec le programme `synthese.c` dans lequel vous aurez commenté les appels aux fonctions non encore développées.

Q. 12 Ajouter à ce fichier le corps de la fonction `afficher`. Tester avec le programme `synthese.c` dans lequel vous aurez commenté les appels aux fonctions non encore développées.

Q. 13 Ajouter à ce fichier le corps de la fonction `afficherPrefixe`. Pour déterminer si le prefixe apparaît au début du nom, vous pouvez utiliser la fonction `strstr` ou la fonction `strncmp`. Tester avec le programme `synthese.c` complet.

5 Bilan

Vous savez maintenant :

- créer un fichier texte ;
- lire un fichier texte ;
- écrire un petit programme manipulant des fichiers textes et des chaîne de caractères.

6 Pour aller plus loin

Vous pouvez améliorer l'exercice de synthèse en :

- ignorant les lignes mal formées dans le répertoire,
- en ajoutant un menu au programme principal pour permettre à l'utilisateur de faire les opérations dans l'ordre qu'il veut,
- en permettant à l'utilisateur de choisir les noms des fichiers contenant le répertoire ou le préfixe à rechercher,
- en ajoutant d'autres fonctionnalités :
  - ajouter, modifier ou supprimer un client,
  - trier le répertoire sur le nom, le prénom ou le numéro,
  - permettre de faire des recherche sur un des champs ou une partie,
  - etc...