TP6 : Gestion des fichiers

1. Partie 1 : Écriture dans un fichier texte avec fputc, fputs et fprintf

Écrivez un programme C qui crée un fichier texte nommé exemple.txt et écrit successivement dans ce fichier :

1. Une phrase écrite caractère par caractère avec la fonction fputc.
2. Une phrase écrite ligne par ligne avec la fonction fputs.
3. Une phrase écrite avec un format personnalisé grâce à fprintf.

Après exécution, ouvrez le fichier exemple.txt dans un éditeur de texte pour observer les différences.

2. Partie 2 : Lecture depuis un fichier texte avec fputc, fputs et fprintf

À partir du fichier exemple.txt créé dans la Partie 1, écrivez un programme C qui lit et affiche successivement :

1. Le contenu du fichier caractère par caractère avec fgetc.
2. Le contenu du fichier ligne par ligne avec fgets.
3. Le contenu du fichier en extrayant des données formatées avec fscanf (par exemple, lire les mots et les afficher séparément).

3. Partie 3 : Lecture et écriture dans un fichier binaire

Vous devez écrire un programme C qui :

1. Définit une structure Etudiant contenant :
   - un tableau nom de 50 caractères,
   - un entier age,
   - un tableau notes de 3 entiers (pour 3 matières).
2. Demande à l'utilisateur combien d'étudiants il souhaite saisir.
3. Alloue dynamiquement un tableau d'étudiants.
4. Lit les informations pour chaque étudiant (nom, âge, notes).
5. Écrit tous les étudiants dans un fichier binaire nommé "etudiants.bin".
6. Lit ensuite ce fichier binaire et affiche les informations de chaque étudiant à l'écran.

4. Partie 4 : Manipulation du curseur de fichier avec ftell et fseek

Vous devez écrire un programme en C qui :

1. Ouvre un fichier texte nommé texte.txt en lecture ("r").
   - Si le fichier n'existe pas, affiche un message d'erreur et arrête le programme.
2. Affiche la taille du fichier en octets en utilisant :

- fseek pour aller à la fin.
- ftell pour obtenir la position actuelle (taille).
- rewind pour revenir au début.

3. Lit et affiche le fichier caractère par caractère avec position :
   - Avant chaque lecture, afficher la position avec ftell.
4. Lit le fichier à l'envers (du dernier caractère au premier) :
   - Utiliser fseek avec SEEK_END et des déplacements négatifs.
   - Afficher chaque caractère.
5. Lit à partir d'une position donnée :
   - Demander à l'utilisateur une position (en octets) à partir du début.
   - Utiliser fseek pour aller directement à cette position et lire la suite du fichier.

Contraintes :

- Gestion des erreurs (if (fp == NULL)).
- Ne pas utiliser de tableaux pour stocker le fichier en entier (lecture directe par curseur).

5. Partie 5 : Gestion des erreurs en lecture de fichiers et allocation mémoire en C

1. Ouverture du fichier :
   - Ouvrez `data.bin` en mode binaire lecture (`"rb"`).
   - Vérifiez si l'ouverture a réussi.
   - En cas d'échec, affichez un message d'erreur clair avec `perror` et quittez le programme.
2. Allocation dynamique du tableau :
   - Allouez un tableau d'entiers pouvant contenir 100 éléments avec malloc.
   - Vérifiez que l'allocation a réussi.
   - En cas d'échec, affichez un message d'erreur clair et fermez le fichier avant de quitter.
3. Lecture du fichier :
   - Utilisez fread pour lire jusqu'à 100 entiers dans le tableau.
   - Vérifiez si le nombre d'entiers lus est inférieur à 100 :
     - Si fin de fichier atteint (feof), affichez un message indiquant que le fichier contient moins de 100 entiers.
     - Si une erreur de lecture survient (ferror), affichez un message d'erreur clair, libérez la mémoire, fermez le fichier et quittez.
4. Affichage des données lues :
   - Affichez les entiers lus à l'écran, en utilisant une fonction dédiée print_array qui prend en paramètre le tableau et le nombre d'éléments lus.
   - Présentez les données proprement (exemple : 10 valeurs par ligne).
5. Libération des ressources :
   - Libérez la mémoire allouée avec free.
   - Fermez le fichier avec fclose.
   - Vérifiez si la fermeture s'est bien déroulée et affichez un message d'erreur en cas de problème.

Conseils techniques

- Chaque fonction critique (fopen, malloc, fread, fclose) doit être suivie d'une vérification.
- Utilisez perror() pour afficher les erreurs liées aux opérations système.
- Utilisez fprintf(stderr, "...") pour afficher les erreurs personnalisées.
- Ne quittez jamais le programme sans avoir libéré la mémoire allouée et fermé le fichier ouvert.
- Testez le programme avec différents fichiers :
  - Un fichier binaire contenant moins de 100 entiers.
  - Un fichier manquant.

6. Partie 6 : Manipulation des fichiers

6.1. Exercice 1

Créez avec votre éditeur un fichier exo1.txt où figureront uniquement votre prénom suivi de votre nom, séparés par un espace. Écrivez une fonction void prenom(FILE *f) qui lit le prénom et l'affiche à l'écran puis en se repositionnant au début du fichier refait la lecture une deuxième fois.

6.1. Exercice 2

Écrivez un programme qui réécrit son propre code dans un autre fichier. Par exemple, si ce programme s'appelle exo_n.c, il doit créer un fichier copie_exo_n.c qui est une copie de lui-même.

Note : *Utilisez la fonction int fputc(char m, FILE *f)* .

6.1. Exercice 3

Écrire un programme C qui permet de remplir un fichier texte à partir des données saisies par l'utilisateur. Après chaque ligne entrée, le programme demande à l'utilisateur de confirmer l'ajout d'une nouvelle ligne (« Voulez vous ajouter une autre ligne dans le fichier texte (O/o N/n).

6.2. Exercice 4

On désire écrire un programme C de gestion de fichier de produits avec un menu général (Création de fichier, Affichage d'un fichier, Recherche d'un produit, Ajout d'un produit, Recherche d'un produit, Ajout d'un produit, Suppression d'un produit, Augmentation de prix).

Pour cela on propose, le fichier «Produit» ayant la structure suivante :

- Référence du produit de type entier
- Désignation de 20 caractères
- Prix unitaire de type réel
- Quantité en stock de type entier
- Quantité seuil de type entier

1. Ecrire une fonction Créer_Produit qui permet de créer séquentiellement le fichier «Produit».

2. Ecrire une fonction Lister_Produit qui permet d'afficher tous les enregistrements du fichier « Produit» (Référence, Désignation, Prix unitaire, Quantité en stock,Quantité seuil) numérotés.
3. Ecrire une fonction Rechercher_Produit qui permet de rechercher un produit dans le fichier par sa référence.
4. Ecrire une fonction Ajouter_Produit qui permet d'ajouter un ou plusieurs produits au fichier « Produit ».
5. Ecrire une fonction Supprimer_Produit qui permet de supprimer un produit dans le fichier « Produit» à travers sa référence. La suppression ne se fait qu'après l'affichage des caractéristiques du produit et la confirmation de l'utilisateur.
6. Ecrire une fonction Augmenter_Prix_Produit qui augmente de 10% le prix unitaire de chaque produit coûtant plus de 100 MAD
7. Ecrire un programme C permettant de Tester les sous programmes ci-dessus et qui affiche un menu.