Lycée TOUCHARD WASHINGTON

Langage C

TD n°10

SNir 1
2021-2022

Accès aux fichiers

Rappel : les fonctions de haut niveau pour l'accès aux fichiers

Opération	fonction	Type de retour	Valeur en cas d'erreur
Ouverture	fopen	FILE *	NULL
Lecture	fread	size_t	Valeur ≠ du nombre d'enregistrements
	fscanf	int	EOF
	fgets	char *	NULL
	fgetc	int	EOF
Écriture	fwrite	size_t	Valeur ≠ du nombre d'enregistrements
	fprintf	int	Valeur négative
	fputs	int	EOF
	fputc	int	EOF
Fermeture	fclose	int	EOF
Déplacement	fseek	int	Valeur négative

Pour ces exercices, il est impératif de tester toutes les valeurs de retour des fonctions d'accès aux fichiers. En cas d'erreur, un traitement approprié doit être effectué ou un message doit s'afficher indiquant le type de l'erreur et le programme doit s'arrêter avec le code d'erreur correspondant.

Exemple

```c
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
int main()
{
    FILE *pFich ;
    char carLu;
    pFich = fopen("fichierTexte.txt","r");
    if(pFich == NULL)
    {
        printf( "%s\n", strerror( errno ) );
        exit(errno);
    }

     return EXIT_SUCCESS;
}
```

1- Lecture d'un fichier TXT

1. Réalisez le codage d'un programme **TD10_Exo1** qui réalise l'affichage du contenu du fichier disponible en ressources **fichierTexte.txt**. La lecture du fichier doit se faire caractère par caractère avec les fonctions de haut niveau.
2. Complétez votre programme pour qu'il affiche le nombre de lettres minuscules non accentuées contenu dans le fichier. Vous devriez trouver : **1832**

2- Taille des fichiers

Soit la définition du type **T_PERSONNE** :

Structure personne

```
#define NB_MAX_CAR 50
typedef struct
{
    char nom[NB_MAX_CAR];
    char prenom[NB_MAX_CAR];
    int age;
    float poids;
    char sexe; // 'f' ou 'm'
} T_PERSONNE;
```

3. Réalisez le codage d'un programme **TD10_Exo2a** qui permet de créer et de sauvegarder dans un fichier de 5 personnes en mode fichier binaire avec la fonction **fwrite** et en mode texte avec la fonction **fprintf**.
4. Quelle devrait être la taille des fichiers générés ?
5. Quelles différences constatez vous dans le contenu des fichiers générés ? vous pourrez vous aider de l'application **ghex** pour voir les contenus de fichier non texte.
6. Quelle est la taille réelle ?
7. Après avoir inséré la ligne #pragma pack(push, 1) à la suite vos lignes d'inclusions et changer le nom des fichiers, relancer votre programme. Compléter le tableau suivant en relevant la taille des fichiers.

		Taille du fichier
Fichier binaire 1	Sans la directive pragma	
Fichier binaire 2	Avec la directive pragma	
Fichier texte	Sans la directive pragma	
Fichier texte	Avec la directive pragma	

Comparez vos fichiers binaires avec la commande Linux **diff**

Pour plus d'explications, voir : https://fr.wikipedia.org/wiki/Alignement_en_m%C3%A9moire

8. Réalisez le codage d'un programme **TD10_Exo2b** qui permet de relire et d'afficher les fichiers de votre voisin. Vous utiliserez pour cela **fscanf** et **fread**.

3- Les fichiers d'image

On donne ici la structure d'un fichier image au format BMP

En-tête fichier BMP

	signature	taille fichier	réservé	adresse relative des informations de l'image
nb octets	2	4 (unsigned int)	4	4

En-tête image

	taille en-tête	largeur image	hauteur image	nombre de plans (toujours = 1)
nb octets	4	4(unsigned int)	4(unsigned int)	2

	Bits utilisés par pixel (1,4,8,16,24 ou 32)	Type de compression 0 = pas de compression 1 = RLE 8bits 2 = RLE 4bits 3 = bitfield	Taille image
nb octets	2	4	4

	Resolution horizontale	Resolution verticale	Nb couleurs utilisées	Nb couleurs importantes
nb octets	4	4	4(unsigned int)	4

	Palette
nb octets	Optionnelle selon le mode

La Palette est un tableau contenant la liste des couleurs.
Pour le mode 8 bits une couleur est codée sur 4 octets : R:G:B:Réservé. La taille de cette palette est donc au maximum de 256 x 4 → 1024 octets.

9. En utilisant les fonctions fseek et fread, réalisez un programme **TD10_EXO3a** permettant d'afficher les dimensions et le nombre de couleurs utilisées d'une image au format BMP.

   Pour tester votre programme, vous utiliserez les images fournies en ressources.

10. Réalisez un nouveau programme **TD10_Exo3b** et ajoutez à votre projet 2 fichiers **image.h** et **image.c** le premier contiendra la définition des structures et les en-têtes de fonctions, le second le code source des fonctions en langage C

11. proposez un type structure "**T_ENTETE_FICHIER_BMP**" pour l'en-tête de fichier BMP et "**T_ENTETE_IMAGE_BMP**" pour l'en-tête de l'image BMP.

12. Réalisez deux fonctions **AfficherEnTeteFichierBMP()** et **AfficherEnTeteImageBMP()**, elles afficheront respectivement le contenu des structures réalisées précédemment et passées en paramètres.

13. Réalisez le programme principal de l'application sous la forme d'un menu. Il demande tout d'abord le choix du fichier BMP, et permet ensuite suivant l'option d'afficher soit l'en-tête du fichier, soit l'en-tête de l'image.