C Programming Language {

[Fichiers]

```
FILE *fopen( const char * filename, const char * mode );
```

}

Fichiers.c

Fichiers.h

FILE *

En C, on manipule un fichier par le biais d'une structure dont le type est FILE et de fonctions qui le prennent en paramètre.

```
FILE *f_handle;
int fputc(int c, FILE *fp);
int fputs(const char *s, FILE *fp);

int fgetc(FILE * fp);
char *fgets(char *buf, int n, FILE *fp);

size_t fread(void *ptr, size_t e_size, size_t nb_e, FILE *a_file);
size_t fwrite(const void *ptr, size_t e_size, size_t nb_e, FILE *a_file);
```

Ouverture

On peut ouvrir un fichier en s'aidant de la fonction fopen, qui retourne un pointeur de FILE:

```
FILE *f_handle;

f_handle = fopen("file.txt", "rb");
```

Chemin vers le fichier visé

Mode d'ouverture du fichier

Fermeture

Quand on a fini d'écrire dans un fichier, on doit le fermer en utilisant la fonction fclose sur la structure FILE ouverte par fopen.

```
fclose(f_handle);
```

Modes d’ouverture des fichiers textes

“r” -> read	Lecture simple
“w” -> write	Écriture depuis le début/création
“a” -> append	Écriture à la fin du fichier ou création
“r+” -> read	Écriture + lecture
“w+” -> write	Écriture depuis le début/lecture/création
“a+” -> append	Écriture à la fin du fichier/lecture/création

Fichiers.c | Fichiers.h

Écrire dans un fichier texte

```c
FILE *f_handle;
f_handle = fopen("file.txt", "w"); // ouvrir ou créer le fichier file.txt

fputs("Hi, mom!", f_handle); // Ajouter une chaîne de caractères
fputc('\n', f_handle); // Ajouter un caractère

fclose(f_handle); // Fermer le fichier
```

Fichiers.c | Fichiers.h

Ajouter à la fin d'un fichier texte

```c
FILE *f_handle;
f_handle = fopen("file.txt", "a"); // ouvrir ou créer le fichier file.txt

fputs("Hi, mom!", f_handle); // Ajouter une chaine de caractères
fputc('\n', f_handle); // Ajouter un caractère

fclose(f_handle); // Fermer le fichier
```

Fichiers.c | Fichiers.h

Lire dans un fichier texte (fgets)

```c
FILE *f_handle;

f_handle = fopen("file.txt", "r");   // Ouvrir ou créer le fichier file.txt


char buff[16];                  //Allouer un espace mémoire où stocker les donnés
char *r = fgets(buff, 16, f_handle); // Lire max 15 char du fichier dans buff.
if (r == NULL) {                // fgets peut échouer: dans ce cas il renvoie NULL
    puts("Reached the end of the file");
}
printf("%s\n", buff); // Hi mom!

```

Fichiers.c | Fichiers.h

Lire dans un fichier texte (fgetc)

```
1
2
3
4   FILE *f_handle;
5
6   f_handle = fopen("file.txt", "r");   // Ouvrir ou créer le fichier file.txt
7
8
9   char c = fgetc(f_handle);           // Lire le caractère suivant dans le fichier
10  printf("%c", buff, c);                  // H
11
12
13
14
```

Fichiers.c | Fichiers.h

Lire dans un fichier texte (fscanf)

```c

FILE *f_handle;
f_handle = fopen("file.txt", "r");   // Ouvrir ou créer le fichier file.txt

char buff[16];                  //Allouer un espace mémoire où stocker les données

fscanf(f_handle, "%s", buff);       // Lire jusqu'au prochain espace dans le fichier
printf("%s\n", buff); // Hi


```

Fichiers.c | Fichiers.h

Lire tout un fichier texte

```c
char buffer[500]; // Alloue un buffer
FILE *f_handle = fopen("text.txt", "r"); // ouvre le fichier en mode lecture



while (fgets(buffer, 500, f_handle)) { // tant que fgets ne retourne pas NULL
    printf("%s", buffer); // afficher la ligne
}

fclose(f_handle); // fermer le fichier
```

Modes d’ouverture de fichier binaire

“rb” -> read	Lecture simple
“wb” -> write	Écriture depuis le début/création
“ab” -> append	Écriture à la fin du fichier ou création
“r+b” -> read	Écriture + lecture
“w+b” -> write	Écriture depuis le début/lecture/création
“a+b” -> append	Écriture à la fin du fichier/lecture/création

Fichiers.c | Fichiers.h

Écrire dans un fichier binaire

```c
// créer deux utilisateurs dans un espace mémoire
User *tim = create_user("Tim", 12);
User *victoria = create_user("Victoria", 31);
User users[2] = {*tim, *victoria};


FILE *f_handle;
f_handle = fopen("file.bin", "wb"); // ouvrir ou créer le fichier file.txt
// écrire l'espace mémoire dans le fichier
size_t written_length = fwrite(users, sizeof(User), 2, f_handle);
fclose(f_handle); // fermer le fichier
```

Fichiers.c

Fichiers.h

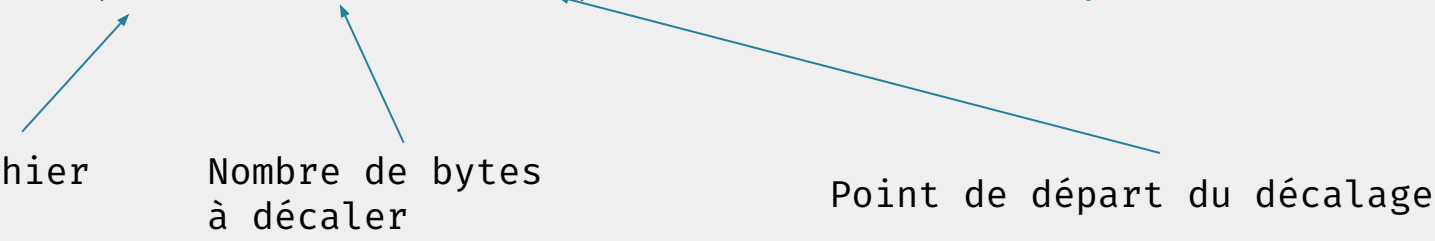
Lire dans un fichier binaire

```c
FILE *f_handle_read;
f_handle_read = fopen("file.bin", "rb"); // ouvrir ou créer le fichier file.txt

User read_users[2]; // Allouer de l'espace pour deux utilisateurs
// Lire dans le fichier sur la taille de deux users
size_t read_length = fread(read_users, sizeof(User), 2, f_handle_read);

// Name: Tim ; Age : 12
printf("read user 1 : \nname: %s\nage : %d\n\n", read_users[0].username, read_users[0].age);
// Name: Victoria ; Age : 31
printf("read user 2 : \nname: %s\nage : %d\n\n", read_users[1].username, read_users[1].age);
fclose(f_handle_read); // Fermer le fichier
```

Système de curseur

Les opérations sur les fichiers se font via un curseur qui commence au début du fichier et se finit à la fin (EOF). Plusieurs techniques permettent de gérer ce curseur.

```c
long cursor = ftell(f_handle); // position du curseur
rewind(f_handle); // Remettre le curseur au début du fichier

fseek(f_handle, -16 ,SEEK_END); // mettre le curseur 16 bytes avant la fin
```



Points de départ de fseek : SEEK_SET(début), SEEK_CUR(actuel), SEEK_END(fin)

Fichiers.c

Fichiers.h

Pratique

Premiers pas

1) Créer un fichier file.txt et le remplir sur les 5 premières lignes de texte
2) Fermer le fichier
3) Rouvrir File.txt et imprimer le contenu du fichier que vous avez créé dans la console

Pratique

Juste à la fin

1) Reprendre le fichier texte créé lors de l'exercice précédent et remplacer les trois dernier caractères par des '*'
2) afficher le résultat

Fichiers.c

Fichiers.h

Pratique

jisonne read

1) Créer une structure qui représente une musique d'une playlist avec un titre et une durée en secondes
2) Créer 5 instances de cette structure
3) Créer un fichier tracks.json qui contient dans la syntaxe json les données des musiques créées précédemment.

Fichiers.c

Fichiers.h

Pratique

jisonne write

1) Réutiliser la structure décrite dans l'exercice précédent
2) Lire un fichier JSON contenant 5 musique et instancier pour chacune de ces musiques dans des structures, en lisant le fichier JSON.

Fichiers.c

Fichiers.h

Pratique

jisone encore

1) Rouvrir le fichier json précédemment créé contenant 5 musiques et
2) Y ajouter 2 musiques sans effacer les précédentes.

Fichiers.c

Fichiers.h

Pratique

bin > jisone

1) Réutiliser la structure de musique utilisée précédemment
2) Cette fois-ci, stocker les instances de cette structures dans un fichier binaire.
3) Fermer le fichier binaire

Fichiers.c

Fichiers.h

Pratique

bin > jisone II : la lecture

1) Lire le fichier binaire précédemment créé et l'afficher dans la console

Pratique

Liste chaînée write

1) Créer une liste chaînée de 30 éléments
2) Stocker cette liste chainée dans un fichier binaire de façon à pouvoir les relire

Fichiers.c

Fichiers.h

Pratique

Liste chaînée read

1) Créer une liste chaînée depuis le fichier binaire stocké précédemment.
2) Afficher chaque noeud pour montrer que la liste a bien été stockée et lue

Fichiers.c

Fichiers.h

Pratique

Playlist chaînée

1) Créer un programme qui permet de créer une liste chaînée dont chaque nœud représente une musique.
2) Cette liste chainée doit être portée par une structure qui contient sa longueur.
3) Instancier 3 de ces listes chaînées et les remplir chacune d'un nombre de musiques différent
4) Stocker toutes ces structures à la suite dans un fichier binaire
5) Lire depuis le fichier les différentes playlists et afficher chacune de leurs musiques

```c
typedef struct {
  char name[16];
  int nb_tracks;
  TrackNode *tracks;
} LinkedPlaylist;
```

```c
typedef struct TrackNode TrackNode;
struct TrackNode {
  char title[16];
  int length;
  TrackNode *next;
}
```