




# **Manipulez des fichiers**



On peut lire et écrire dans des fichiers en langage C. Ces fichiers seront écrits sur le disque dur de votre ordinateur : ils restent là même si vous arrêtez le programme ou l'ordinateur.

Pour lire et écrire dans des fichiers, nous allons nous servir de fonctions situées dans des bibliothèques standard.

Incluez au moins les bibliothèques **stdio.h** et **stdlib.h** en haut de votre **fichier .c**

### ► Ouvrez un fichier avec **fopen**

Chaque fois que vous voulez ouvrir un fichier, que ce soit pour le lire ou pour y écrire, il faut :

1. Appeler la fonction d'ouverture de fichier **fopen** qui renvoie un pointeur sur le fichier.
2. Vérifier si l'ouverture a réussi (c'est-à-dire si le fichier existait) en testant la valeur du pointeur qu'on a reçu.

**FILE\* fopen(const char\* nomDuFichier, const char\* modeOuverture);**

Cette fonction attend deux paramètres :

1. Le nom du fichier à ouvrir.
2. Le mode d'ouverture du fichier, c'est-à-dire une indication qui mentionne ce que vous voulez faire : seulement écrire dans le fichier, seulement le lire, ou les deux à la fois.

### Remarque

Cette fonction renvoie un pointeur sur **FILE** (une structure de type **FILE** qui est définie dans **stdio.h**). Elle renvoie un **FILE\***.

Créons donc un pointeur de **FILE** au début de notre fonction, par exemple la fonction **main** :

```
int main(int argc, char *argv[])
{
    FILE* fichier = NULL;

    return 0;
}
```



Il y a un code à envoyer qui indiquera à l'ordinateur si vous ouvrez le fichier en mode de lecture seule, d'écriture seule, ou des deux à la fois.

Voici les principaux modes d'ouverture possibles :

1. **"r"** : lecture seule. Vous pourrez lire le contenu du fichier, mais pas y écrire. Le fichier doit avoir été créé au préalable.
2. **"w"** : écriture seule. Vous pourrez écrire dans le fichier, mais pas lire son contenu. Si le fichier n'existe pas, il sera créé.
3. **"a"** : mode d'ajout. Vous écrirez dans le fichier, en partant de la fin du fichier. Vous ajouterez donc du texte à la fin du fichier. Si le fichier n'existe pas, il sera créé. Ce mode d'ajout peut être utile si vous voulez seulement ajouter des informations à la fin du fichier.
4. **"a+"** : ajout en lecture / écriture à la fin. Vous écrivez et lisez du texte à partir de la fin du fichier. Si le fichier n'existe pas, il sera créé.
5. **"r+"** : lecture et écriture. Vous pourrez lire et écrire dans le fichier. Le fichier doit avoir été créé au préalable.
6. **"w+"** : lecture et écriture, avec suppression du contenu au préalable. Le fichier est donc d'abord vidé de son contenu, vous pouvez y écrire, et le lire ensuite. Si le fichier n'existe pas, il sera créé.

Le code suivant ouvre le fichier test.txt en mode "r+" (lecture et écriture) :

```
int main(int argc, char *argv[])
{
    FILE* fichier = NULL;

    fichier = fopen("test.txt", "r+");

    return 0;
}
```

Le pointeur **fichier** devient alors un pointeur sur **test.txt**

Où doit être situé test.txt ?

Il doit être situé dans le même dossier que votre exécutable ( .exe ). Pour les besoins de ce chapitre, créez un fichier test.txt dans le même dossier que le .exe

**fichier = fopen("dossier/test.txt", "r+");**

le fichier test.txt est dans un sous-dossier appelé dossier

**Remarque:** Juste après l'ouverture du fichier, il faut impérativement vérifier si l'ouverture a réussi ou non. Pour faire ça, c'est très simple : si le pointeur vaut NULL , l'ouverture a échoué. S'il vaut autre chose que NULL , l'ouverture a réussi.

```
int main(int argc, char *argv[])
{
    FILE* fichier = NULL;

    fichier = fopen("test.txt", "r+");

    if (fichier != NULL)
    {
        // On peut lire et écrire dans le fichier
    }
    else
    {
        // On affiche un message d'erreur si on veut
        printf("Impossible d'ouvrir le fichier test.txt");
    }

    return 0;
}
```

Faites toujours cela lorsque vous ouvrez un fichier. Si vous ne le faites pas et que le fichier n'existe pas, vous risquez un plantage du programme par la suite.



## ➤ Fermez un fichier avec `fclose`

Une fois que vous aurez fini de travailler avec le fichier, il faudra le « fermer ». On utilise pour cela la fonction **`fclose`** qui a pour rôle de libérer la mémoire, c'est-à-dire supprimer votre fichier chargé dans la mémoire vive.

```
int fclose(FILE* pointeurSurFichier);
```


Elle renvoie un `int` qui indique si elle a réussi à fermer le fichier. Ce `int` vaut :

**0** : si la fermeture a marché ;

**EOF** : si la fermeture a échoué. **EOF** est un `define` situé dans **`stdio.h`** qui correspond à un nombre spécial, utilisé pour dire soit qu'il y a eu une erreur, soit que nous sommes arrivés à la fin du fichier. Dans le cas présent, cela signifie qu'il y a eu une erreur.

Il faut toujours penser à fermer son fichier une fois que l'on a fini de travailler avec. Cela permet de libérer de la mémoire.

Si vous oubliez de libérer la mémoire, votre programme risque à la fin de prendre énormément de mémoire qu'il n'utilise plus.



```
int main(int argc, char *argv[])
{
    FILE* fichier = NULL;

    fichier = fopen("test.txt", "r+");

    if (fichier != NULL)
    {
        // On lit et on écrit dans le fichier

        // ...

        fclose(fichier); // On ferme le fichier qui a été ouvert
    }

    return 0;
}
```

## ► Écrivez dans un fichier

Les trois fonctions que nous allons étudier :

1. **fputc** écrit un caractère dans le fichier (UN SEUL caractère à la fois).
2. **fputs** écrit une chaîne dans le fichier.
3. **fprintf** écrit une chaîne formatée dans le fichier.



## ➤ Écrivez un caractère dans le fichier avec fputc

```
int main(int argc, char *argv[])
{
    FILE* fichier = NULL;

    fichier = fopen("test.txt", "w");

    if (fichier != NULL)
    {
        fputc('A', fichier); // Écriture du caractère A
        fclose(fichier);
    }

    return 0;
}
```

## ➤ Écrivez une chaîne dans le fichier avec fputs

```
int main(int argc, char *argv[])
{
    FILE* fichier = NULL;

    fichier = fopen("test.txt", "w");

    if (fichier != NULL)
    {
        fputs("Salut les développeurs\nBienvenue sur OpenClassrooms !", fichier);
        fclose(fichier);
    }

    return 0;
}
```

## ➤ Écrivez une chaîne "formatée" dans le fichier avec fprintf

```
int main(int argc, char *argv[])
{
    FILE* fichier = NULL;
    int age = 0;

    fichier = fopen("test.txt", "w");

    if (fichier != NULL)
    {
        // On demande l'âge
        printf("Quel âge avez-vous ? ");
        scanf("%d", &age);

        // On l'écrit dans le fichier
        fprintf(fichier, "Le Monsieur qui utilise le programme, il a %d ans", age);
        fclose(fichier);
    }

    return 0;
}
```

## ➤ Lisez dans un fichier

**int fgetc(FILE\* pointeurDeFichier);**

Nous pouvons utiliser quasiment les mêmes fonctions que pour l'écriture, le nom change juste un petit peu :

1. `fgetc` lit un caractère.
2. `fgets` lit une chaîne.
3. `fscanf` lit une chaîne "formatée".

### ► Lisez un caractère avec `fgetc`

**`fgetc`** avance le curseur d'un caractère à chaque fois que vous en lisez un. Si vous appelez **`fgetc`** une seconde fois, la fonction lira donc le second caractère, puis le troisième et ainsi de suite.

```
int main(int argc, char *argv[])
{
    FILE* fichier = NULL;
    int caractereActuel = 0;

    fichier = fopen("test.txt", "r");

    if (fichier != NULL)
    {
        // Boucle de lecture des caractères un à un
        do
        {
            caractereActuel = fgetc(fichier); // On lit le caractère
            printf("%c", caractereActuel); // On l'affiche
        } while (caractereActuel != EOF); // On continue tant que fgetc n'a pas retourné EOF (fin de
fichier)

        fclose(fichier);
    }

    return 0;
}
```

## ► Lisez une chaîne avec fgets

Cette fonction lit une chaîne dans le fichier. Ça vous évite d'avoir à lire tous les caractères un par un. La fonction lit au maximum une ligne (elle s'arrête au premier \n qu'elle rencontre). Si vous voulez lire plusieurs lignes, il faudra faire une boucle.

**char\* fgets(char\* chaine, int nbreDeCaracteresALire, FILE\* pointeurSurFichier);**

```
#define TAILLE_MAX 1000 // Tableau de taille 1000

int main(int argc, char *argv[])
{
    FILE* fichier = NULL;
    char chaine[TAILLE_MAX] = ""; // Chaîne vide de taille TAILLE_MAX

    fichier = fopen("test.txt", "r");

    if (fichier != NULL)
    {
        fgets(chaine, TAILLE_MAX, fichier); // On lit maximum TAILLE_MAX caractères du fichier, on stocke le
        tout dans "chaine"
        printf("%s", chaine); // On affiche la chaîne

        fclose(fichier);
    }

    return 0;
}
```

La différence, c'est qu'ici on ne fait pas de boucle. On affiche toute la chaîne d'un coup. Vous aurez sûrement remarqué maintenant l'intérêt que peut avoir un **#define** dans son code pour définir la taille maximale d'un tableau, par exemple. En effet, **TAILLE\_MAX** est ici utilisé à deux endroits du code :

1. une première fois pour définir la taille du tableau à créer ;
2. une autre fois dans le `fgets` pour limiter le nombre de caractères à lire.

L'avantage ici, c'est que si vous vous rendez compte que la chaîne n'est pas assez grande pour lire le fichier, vous n'avez qu'à changer la ligne du `define` et recompiler.

```
#define TAILLE_MAX 1000

int main(int argc, char *argv[])
{
    FILE* fichier = NULL;
    char chaine[TAILLE_MAX] = "";

    fichier = fopen("test.txt", "r");

    if (fichier != NULL)
    {
        while (fgets(chaine, TAILLE_MAX, fichier) != NULL) // On lit le fichier tant qu'on ne reçoit pas
d'erreur (NULL)
        {
            printf("%s", chaine); // On affiche la chaîne qu'on vient de lire
        }

        fclose(fichier);
    }

    return 0;
}
```

## ► Lisez une chaîne "formatée" avec fscanf

Supposons que votre fichier contienne trois nombres séparés par un espace, qui sont par exemple les trois plus hauts scores obtenus à votre jeu : 15 20 30

Vous voudriez récupérer chacun de ces nombres dans une variable de type int .

La fonction **fscanf** va vous permettre de faire ça rapidement.

```
int main(int argc, char *argv[])
{
    FILE* fichier = NULL;
    int score[3] = {0}; // Tableau des 3 meilleurs scores

    fichier = fopen("test.txt", "r");

    if (fichier != NULL)
    {
        fscanf(fichier, "%d %d %d", &score[0], &score[1], &score[2]);
        printf("Les meilleurs scores sont : %d, %d et %d", score[0], score[1], score[2]);

        fclose(fichier);
    }

    return 0;
}
```