

Rappels sur la manipulation des fichiers en langage C (via les pointeurs sur les fichiers)

Dans ce document sont rappelées les notions fondamentales quant à la manipulation des fichiers en langage C.

En C, un fichier est une suite d'octets. Les informations contenues dans un fichier ne sont pas forcément du même type (il peut y avoir un char, un int, une struct, etc.). Les opérations disponibles en C pour manipuler des fichiers sont les suivantes : création, ouverture, fermeture, lecture, écriture, destruction, renommage. La plupart des fonctions permettant de manipuler les fichiers sont dans la bibliothèque standard `stdio.h`.

Déclaration d'un fichier

```
FILE *f1 ;
```

On définit un pointeur `f1` vers un fichier (à noter que “`FILE`” doit impérativement être en majuscules). Ce pointeur fournit l'adresse d'un enregistrement du fichier.

Remarquons qu'il existe des fichiers spéciaux : la sortie standard est le fichier `stdout` et l'entrée standard `stdin`.

Ouverture d'un fichier

```
FILE *fopen(char *nom, char *mode) ;
```

Il y a deux chaînes de caractères en argument (`nom` et `mode`). La chaîne `nom` correspond au chemin du fichier que l'on souhaite ouvrir. Par exemple : sous Windows,

```
"a:\toto.dat"
```

Quant à la chaîne `mode`, il s'agit du mode d'accès aux fichiers. Pour des fichiers texte :

- “`r`” : lecture seule ;
- “`w`” : écriture seule (destruction de l'ancienne version si elle existe) ;
- “`w+`” : lecture/écriture (destruction de l'ancienne version si elle existe) ;
- “`r+`” : lecture/écriture d'un fichier existant (mise à jour). Pas de création d'une nouvelle version ;
- “`a+`” : lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version, le pointeur (de lecture/écriture) est positionné à la fin du fichier.

Pour les fichiers binaires :

- “`rb`” : lecture seule ;
- “`wb`” : écriture seule (destruction de l'ancienne version si elle existe) ;
- “`wb+`” : lecture/écriture (destruction de l'ancienne version si elle existe) ;
- “`rb+`” : lecture/écriture d'un fichier existant (mise à jour). Pas de création d'une nouvelle version ;
- “`ab+`” : lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version, le pointeur (de lecture/écriture) est positionné à la fin du fichier.

À l'ouverture, le pointeur est positionné au début du fichier (sauf pour les modes “`a+`” et “`ab+`”). Par exemple,

```
FILE *fichier ;  
fichier=fopen("toto.dat","rb") ;
```

Fermeture d'un fichier

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

Retourne 0 si la fermeture s'est bien passée, EOF en cas d'erreur (par exemple, la fermeture d'un fichier non préalablement ouvert ...). Il faut toujours fermer un fichier à la fin de son utilisation. Par exemple :

```
FILE *fichier ;  
fichier=fopen("/tata/tutu/toto.txt","r") ;  
/* Ici les instructions de traitements ... */  
fclose (fichier) ;
```

Destruction d'un fichier

```
int remove (char *nom) ;
```

Retourne 0 si destruction s'est bien passée. Par exemple :

```
remove("/tata/tutu/toto.txt") ;
```

Renommage d'un fichier

```
int rename(char *oldname, char *newname) ;
```

Retourne 0 si le renommage s'est correctement passé.

Positionnement du pointeur au début du fichier

```
void rewind (FILE *fichier) ;
```

Évidemment il faut que le fichier ait déjà été ouvert.

Écriture dans un fichier

```
int fputc (char c, FILE *fichier) ;
```

Écrit la valeur de c à la position courante du pointeur, puis le pointeur avance d'une case mémoire. Par exemple,

```
fputc ('A',fichier) ;
```

On peut également écrire une chaîne de caractères dans un fichier par `fprintf(FILE *fichier, char *format, ...)` ;

Par exemple,

```
int x ;
x=5 ;
fprintf (fichier, "x=%d et %s", x, "erreur de calcul") ;
```

permet d'écrire la chaîne "x=5 et erreur de calcul" dans le fichier (sur lequel pointe) `fichier`.

Lecture dans un fichier

```
int fgetc (FILE *fichier) ;
```

Lit un caractère, mais retourne un entier (retourne EOF si erreur en fin de fichier) puis le pointeur avance d'une case. Par exemple,

```
char c ;
c=(char)fgetc(fichier) ;
```

On peut également lire une chaîne de caractères. Pour cela on utilise

```
int fscanf(FILE *fichier, char *format, ...)
```

Le troisième argument est un pointeur. Si on utilise comme format `"%as"` alors on récupère une chaîne de caractères jusqu'à un espace ou un saut de ligne, et l'option `a` permet d'allouer la mémoire nécessaire pour stocker la chaîne trouvée. La valeur renvoyée est EOF si la fonction a échoué. Évidemment le pointeur se déplace à la chaîne suivante. Par exemple,

```
FILE *f ;
f=fopen("/toto.txt", "r") ;
char *buffer ;
fscanf(f, "%as", &buffer) ;
```

Dans cet exemple, la chaîne lue par le `fscanf` est stockée dans la chaîne `buffer`.

Déplacement dans un fichier

Les fonctions utilisées jusqu'ici parcourent un fichier séquentiellement du début à la fin, en déplaçant implicitement un curseur dans le fichier. On peut jouer avec ce curseur grâce à la fonction `int fseek(FILE *f, long decalage, int origine)`. La valeur renvoyée est 0 en cas de succès, autre chose sinon. Cette fonction permet de se positionner en `origine + decalage`. Le paramètre `decalage` est exprimé en octets et `origine` peut prendre les 3 valeurs suivantes :

- `SEEK_SET` : début du fichier ;
- `SEEK_CUR` : position courante ;
- `SEEK_END` : fin du fichier ;

Pour connaître la position courante à partir du début du fichier, on utilise `long ftell(FILE *f)`.