Programmation C

Séance 4: L'accès aux fichiers

Fichiers

Le C offre la possibilité de lire et d'écrire des données dans un fichier.
Pour des raisons d'efficacité, les accès à un fichier se font par l'intermédiaire d'une mémoire-tampon (*buffer*), ce qui permet de réduire le nombre d'accès aux périphériques (disque...).
Pour pouvoir manipuler un fichier, un programme a besoin d'un certain nombre d'informations : l'adresse de l'endroit de la mémoire-tampon où se trouve le fichier, la position de la tête de lecture, le mode d'accès au fichier (lecture ou écriture) ...Ces informations sont rassemblées dans une structure dont le type, FILE *, est défini dans stdio.h. Un objet de type FILE * est appelé *flot de données*.

La fonction fopen

Cette fonction, de type `FILE*` ouvre un fichier et lui associe un flot de données. Sa syntaxe est : `fopen("nom-de-fichier","mode")`.
La valeur retournée par `fopen` est un flot de données. Si l'exécution de cette fonction ne se déroule pas normalement, la valeur retournée est le pointeur `NULL`. Il est donc recommandé de toujours tester si la valeur renvoyée par la fonction `fopen` est égale à `NULL` afin de détecter les erreurs (lecture d'un fichier inexistant...).
Le premier argument de `fopen` est le nom du fichier concerné, fourni sous forme d'une chaîne de caractères. Le second argument, *`mode`*, est une chaîne de caractères qui spécifie le mode d'accès au fichier.
Les différents modes d'accès pour un fichier texte sont les suivants :

`"r"`	ouverture d'un fichier texte en lecture
`"w"`	ouverture d'un fichier texte en écriture
`"a"`	ouverture d'un fichier texte en écriture à la fin
`"r+"`	ouverture d'un fichier texte en lecture/écriture
`"w+"`	ouverture d'un fichier texte en lecture/écriture
`"a+"`	ouverture d'un fichier texte en lecture/écriture à la fin

- Si le mode contient la lettre r, le fichier doit exister.
- Si le mode contient la lettre w, le fichier peut ne pas exister. Dans ce cas, il sera créé. Si le fichier existe déjà, son ancien contenu sera perdu.
- Si le mode contient la lettre a, le fichier peut ne pas exister. Dans ce cas, il sera créé. Si le fichier existe déjà, les nouvelles données seront ajoutées à la fin du fichier précédent.

La fonction fclose

Elle permet de fermer le flot qui a été associé à un fichier par la fonction `fopen`. Sa syntaxe est : `fclose(`*`flot`*`)` où *`flot`* est le flot de type `FILE*` retourné par la fonction `fopen` correspondant.
La fonction `fclose` retourne un entier qui vaut zéro si l'opération s'est déroulée normalement (et une valeur non nulle en cas d'erreur).
Pour les opérations d'entrée/sortie sur des fichiers on peut employer les fonctions *fscanf* et *fprintf*. Elles sont identiques a *scanf* et *printf*, mis à part que le premier argument est un pointeur de fichier qui précise le fichier à lire ou à écrire.

Le programme suivant crée le ficher essaifichier.txt dans votre dossier et il y écrit le mot Bonjour :

```
#include <stdio.h>

main()
{
	FILE *fid;	//pointeur sur un FILE

	fid=fopen("essaifichier.txt", "w");	//écrire dans le FILE fid
	fprintf(fid,"Bonjour");
	fclose(fid);				//libérer le pointeur de fichier
}
```

Exercice 1

Ecrire la procédure *ecrireEntiers()* qui lit au clavier des entiers positifs (la saisie s'arrête lorsqu'on entre une valeur négative) et les écrit dans le fichier *entiers.txt*.
Ecrire la procédure *lireEntiers()* dans laquelle vous ouvrez en lecture le fichier *entiers.txt*, lisez et affichez son contenu à l'écran.
Tester ces procédures dans la fonction *main()*.

Exercice 2 : Lecture par caractère

Écrivez une fonction compte_car (FILE * f) qui renvoie le nombre de caractères d'un fichier.
Écrivez une fonction compte_ligne (FILE * f) qui renvoie le nombre de lignes.

Indications :
int fgetc(FILE *f) : retourne le code ASCII du caractère qui a été lu.
Si la fonction n'a pas pu lire de caractère, elle retourne EOF.
EOF : constante 'fin de fichier' définie dans <stdio.h>
int feof(FILE *f) indique si on est en fin de fichier ou non (0).

Exercice 3 : Écriture par caractère

La fonction int fputc(char c, FILE * f) écrit le caractère c dans le fichier f.
Écrivez un programme qui se ré-écrit son code dans un autre fichier (crée une copie de lui-même).