Ecole supérieure de la Statistique et de
l'Analyse de l'Information de Tunis

Algorithmique et programmation II
1ère année Ingénieur
aicha.elgolli@essai.ucar.tn

La gestion des fichiers

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 fichiers permettent d'inscrire des informations sur le support disque et permettent par la suite de les récupérer, les modifier et/ou les supprimer.
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. 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* (en anglais, stream). Avant de lire ou d'écrire dans un fichier, on notifie son accès par la commande fopen. Cette fonction prend comme argument le nom du fichier, négocie avec le système d'exploitation et initialise un flot de données, qui sera ensuite utilisé lors de l'écriture ou de la lecture. Après les traitements, on annule la liaison entre le fichier et le flot de données grâce à la fonction fclose.

Déclaration d'un fichier

FILE *fl ;

On définit un pointeur fl 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 et fermeture d'un fichier

La fonction fopen

Cette fonction, de type FILE* ouvre un fichier et lui associe un flot de données. Sa syntaxe est:

FILE *fopen(char *nom, char *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. On préférera définir le nom du fichier par une constante symbolique au moyen de la directive #define plutôt que d'expliciter le nom de fichier dans le corps du programme. Le second argument, mode, est une chaîne de caractères qui spécifie le mode d'accès au fichier. Les spécificateurs de mode d'accès différent suivant le type de fichier considéré. On distingue
– les fichiers textes, pour lesquels les caractères de contrôle (retour à la ligne ...) seront interprétés en tant que tels lors de la lecture et de l'écriture;
– les fichiers binaires, pour lesquels les caractères de contrôle ne sont pas interprétés.
Les différents modes d'accès sont les suivants:
Pour des fichiers texte :

- "r" : lecture seule ; position au début du fichier et erreur si le fichier n'existe pas.
- "w" : écriture seule, position au début du fichier (destruction de l'ancienne version si elle existe). Si le fichier n'existe pas il y a création;
- "a" : écriture seule à la fin du fichier, si fichier inexistant création ;
- "w+" : lecture/écriture en début du fichier (destruction de l'ancienne version si elle existe) ;
- "r+" : lecture/écriture d'un fichier existant (mise à jour) et en début du fichier. Pas de création d'une nouvelle version erreur si fichier inexistant ;
- "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 les formats deviennent : ab, ab+, rb, rb+, wb, wb+.
Ces modes d'accès ont pour particularités:

- 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.

Par exemple,

FILE *fichier ;
fichier=fopen("toto.dat","rb") ;
Trois flots standards peuvent être utilisés en C sans qu'il soit nécessaire de les ouvrir ou de les fermer:
– stdin (standard input): unité d'entrée (par défaut, le clavier);
– stdout (standard output): unité de sortie (par défaut, l'écran);
– stderr (standard error): unité d'affichage des messages d'erreur (par défaut, l'écran).
Il est fortement conseillé d'afficher systématiquement les messages d'erreur sur stderr afin que ces messages apparaissent à l'écran même lorsque la sortie standard est redirigée.

La fonction fclose

int fclose (FILE *fichier) ;

Elle permet de fermer le flot qui a été associé à un fichier par la fonction fopen. 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 : 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) ;

Entrées-sorties formatées

Les entrées-sortie formatées sont assurées par les fonctions ***fprintf*** et ***fscanf***.

a) ***La fonction fprintf***

La fonction utilisée pour l'écriture dans un fichier est fprintf.

int fprintf(FILE* pointeurFichier, char* format,var1, var2,...) ;

cette fonction écrit les données var_i dans le flux pointeurFichier en respectant le format spécifié par la chaîne format. Elle retourne le nombre d'octets écrits sur le flux.
Par exemple,

FILE *fichier ;
fichier=fopen("log.txt","a") ;
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 « log.txt » (sur lequel pointe) fichier.

b) ***La fonction fscanf***

La fonction utilisée pour la lecture dans un fichier est fscanf.

int fscanf(FILE* pointeurFichier, char* format, adrvar1, adrvar2,...) ;

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.
Evidemment 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.

Impression et lecture de caractères

La manipulation de fichiers avec les instructions **fprintf** et **fscanf** n'est pas assez flexible pour manipuler de façon confortable des textes écrits. Il est alors avantageux de traiter le fichier séquentiellement caractère par caractère.

a) Ecrire un caractère dans un fichier séquentiel - fputc

fputc(<C> , <FP>);

fputc transfère le caractère indiqué par <C> dans le fichier référencé par <FP> et avance la position de la tête de lecture/écriture au caractère suivant.

* représente un caractère (valeur numérique de 0 à 255) ou le symbole de fin de fichier **EOF**
* **<FP>** est un pointeur du type **FILE*** qui est relié au nom du fichier cible.

Remarque
L'instruction
fputc('a', stdout); est identique à **putchar('a');**

b) Lire un caractère dans un fichier séquentiel - fgetc

Ecole supérieure de la Statistique et de
l'Analyse de l'Information de Tunis

<C> = fgetc(<FP>);

fgetc fournit comme résultat le prochain caractère du fichier référencé par <FP> et avance la position de la tête de lecture/écriture au caractère suivant. A la fin du fichier, **fgets** retourne **EOF.**

<C> représente une variable du type **int** qui peut accepter une valeur numérique de 0 à 255 ou le symbole de fin de fichier **EOF.**

<FP> est un pointeur du type **FILE*** qui est relié au nom du fichier à lire.

Remarque

L'instruction

C = fgetc(stdin); est identique à **C = getchar();**

Détection de la fin d'un fichier séquentiel

Lors de la fermeture d'un fichier ouvert en écriture, la fin du fichier est marquée automatiquement par le symbole de fin de fichier **EOF** *(End Of File)*. Lors de la lecture d'un fichier, la fonction **feof(<FP>)** nous permettent de détecter la fin du fichier:

feof(<FP>);

feof retourne une valeur différente de zéro, si la tête de lecture du fichier référencé par <FP> est arrivée à la fin du fichier; sinon la valeur du résultat est zéro.

<FP> est un pointeur du type **FILE*** qui est relié au nom du fichier à lire.

Pour que la fonction **feof** détecte correctement la fin du fichier, il faut qu'après la lecture de la dernière donnée du fichier, la tête de lecture arrive jusqu'à la position de la marque **EOF**. Nous obtenons cet effet seulement si nous terminons aussi la chaîne de format de **fscanf** par un retour à la ligne '\n' (ou par un autre signe d'espacement).

Exemple

Une boucle de lecture typique pour lire les enregistrements d'un fichier séquentiel référencé par un pointeur FP peut avoir la forme suivante:

```
while (!feof(FP))
  {
      fscanf(FP, "%s\n ... \n", NOM, ... );
  ...
  }
```

Exemple

Le programme suivant lit et affiche le fichier "C:\AUTOEXEC.BAT" en le parcourant caractère par caractère:

```
#include <stdio.h>
#include <stdlib.h>
main()
{
 FILE *FP;
 FP = fopen("C:\\AUTOEXEC.BAT", "r");
 if (!FP)
   {
    printf("Impossible d'ouvrir le fichier\n");
    exit(-1);
   }
 while (!feof(FP))
    putchar(fgetc(FP));
 fclose(FP);
 return 0;
}
```

Dans une chaîne de caractères constante, il faut indiquer le symbole '\' (back-slash) par '\\', pour qu'il ne soit pas confondu avec le début d'une séquence d'échappement (p.ex: \n, \t, \a, ...).

EXERCICE

On se propose de créer un fichier « INFORM.TXT » qui est formé d'enregistrements contenant comme information le matricule, le nom et prénom d'une personne. Chaque enregistrement est donc constitué de 3 rubriques. L'utilisateur doit entrer au clavier le nombre de personnes et les infos des personnes. Le programme se chargera de créer le fichier correspondant sur disque dur.
Après avoir écrit et fermé le fichier, le programme va rouvrir le même fichier en lecture et afficher son contenu, sans utiliser le nombre d'enregistrements introduit dans la première partie.

```
#include <stdio.h>
#include <stdlib.h>
main()
{
 /* Déclarations : */
 /* Nom du fichier et pointeur de référence */
 char NOM_FICH[] = "INFORM.TXT";
 FILE *FICHIER;
  /* Autres variables */
 char NOM[30], PRENOM[30];
 int MATRICULE;
 int I,N_ENR;

 /* Ouverture du nouveau fichier en écriture */
 FICHIER = fopen(NOM_FICH, "w");
 if (!FICHIER)
   {
    printf("\aERREUR: Impossible d'ouvrir "
         "le fichier: %s.\n", NOM_FICH);
    exit(-1);
   }
  /* Saisie des données et création du fichier */
 printf("*** Création du fichier %s ***\n", NOM_FICH);
 printf("Nombre d'enregistrements à créer : ");
 scanf("%d",&N_ENR);
 for (I=1; I<=N_ENR; I++)
  {
   printf("Enregistrement No: %d \n", I);
   printf("Numéro de matricule : ");
   scanf("%d",&MATRICULE);
   printf("Nom    : ");
   scanf("%s",NOM);
   printf("Prénom : ");
   scanf("%s",PRENOM);
   fprintf(FICHIER, "%d\n%s\n%s\n", MATRICULE,
NOM, PRENOM);
   }
 /* Fermeture du fichier */
 fclose(FICHIER);

 /* Ouverture du fichier en lecture */
 FICHIER = fopen(NOM_FICH, "r");
 if (!FICHIER)
   {
    printf("\aERREUR: Impossible d'ouvrir "
         "le fichier: %s.\n", NOM_FICH);
    exit(-1);
   }
  /* Affichage du fichier */
 printf("*** Contenu du fichier  %s ***\n", NOM_FICH);
 while (!feof(FICHIER))
  {
   fscanf(FICHIER, "%d\n%s\n%s\n", &MATRICULE,
NOM, PRENOM);
   printf("Matricule : %d\n", MATRICULE);
   printf("Nom et prénom : %s %s\n", NOM, PRENOM);
   }
 /* Fermeture du fichier */
 fclose(FICHIER);
 return 0; }
```