LES FICHIERS EN C

Dr Youssou KASSE

Université Alioune Diop de Bambey

Youssou.kasse@uadb.edu.sn

Généralités

- Les données d’un programme stockées en mémoire centrale sont perdues dès la fin de l'exécution du programme.
- Un fichier est une structure externe (stockée sur disque, disquette, bande…) qui permet de conserver durablement des informations mais au prix d'un temps d'accès très supérieur par rapport à la mémoire centrale.

Généralités

- Le langage C offre la possibilité de lire et d'écrire des données dans un fichier.
- Mais, avant de lire ou d'écrire dans un fichier, celui-ci doit être ouvert.
- Après les traitements, le fichier doit être fermé.
- En langage C, on manipule un fichier grâce à un objet de type **FILE*** appelé *flot de données* (en anglais, stream).

Ouverture d'un fichier

- Avant d'ouvrir un fichier il faut d'abord déclarer un flot:
  - FILE* nom_flot;
- Pour ouvrir le fichier on fait appel à la fonction **fopen:**
  - nom_flot = fopen(nom_fichier, mode_d_ouverture);
- La fonction fopen() retourne un objet de type FILE* si l'ouverture s'est déroulée convenablement; dans le cas contraire elle retourne le pointeur **NULL** (fichier inexistant, fichier protégé, ...).
- Les arguments de **fopen** sont des chaînes de caractères. La première est le ***nom du fichier***, la seconde est le ***mode d'ouverture*** qui indique les opérations qui pourront être effectuées sur le fichier (lecture, écriture ou ajout de données).

Modes d'ouverture d'un fichier

mode	signification
"r"	Ouvre un fichier existant pour la lecture
"w"	Crée et ouvre un fichier pour l'écriture. Si le fichier existe déjà, il sera écrasé.
"a"	Ouvre un fichier en ajout. L'ajout se fait à la fin d'un fichier existant. Si le fichier n'existe pas, il sera créé.
"r+"	Ouvre un fichier existant pour la lecture et l'écriture
"w+"	Crée et ouvre un fichier existant pour la lecture et l'écriture. Si le fichier existe déjà, il sera écrasé
"a+"	Ouvre un fichier pour la lecture et l'ajout de données à la fin d'un fichier existant. Si le fichier n'existe pas, il sera créé.
["r+b" ou "rb+]	ouverture d'un fichier binaire en lecture/écriture.

Remarque

- Il est recommandé de toujours tester la valeur de retour de la fonction **fopen** afin de détecter une éventuelle erreur d'ouverture.

```
#include <stdio.h>
int main()  {
FILE *fp;
fp = fopen ("c:\\algo\\essai.txt", "r");
if (fp == NULL)
{
printf("Impossible d'ouvrir le fichier");
return 0;
}
...
}
```

Fermeture d'un fichier

- Tout fichier ouvert doit être fermé après son utilisation.
- La fonction **fclose( )** permet de fermer le fichier repéré par un flot selon la syntaxe suivante:
  - fclose(nom_flot);

Exemple :

```
FILE * fp;
fp = fopen ("C:\\algo\\essai.txt", "w");
...
fclose(fp);
```

Les voies de communication standard

- Quand un programme est lancé par le système, celui-ci ouvre trois fichiers correspondant aux trois voies de communication standard : *standard input*, *standard output* et *standard error*.
- Il y a trois constantes prédéfinies dans stdio.h de type pointeur vers FILE qui repèrent ces trois fichiers. Elles ont pour nom respectivement stdin(par défaut, le clavier), stdout (par défaut, l'écran) et stderr(par défaut, l'écran).

Lecture et écriture formatés

- La fonction **fprintf** permet d'écrire dans un fichier. Elle s'utilise exactement de la même manière que **printf** mais en ajoutant un flot comme premier paramètre.
- Sa syntaxe est la suivante:

fprintf(nom_flot, "*chaîne de contrôle*", expression1,..., expressionN);

```
#include <stdio.h>
 FILE *fp;
if ((fp = fopen("donnees","r")) == NULL)
 {
fprintf(stderr,"Impossible d'ouvrir le fichier données en
lecture\n");
 exit(1);
 }
```

Lecture et écriture formatés

- La fonction **fscanf** permet de lire à partir d'un fichier. Elle s'utilise exactement de la même manière que **scanf** mais en ajoutant un flot comme premier paramètre.
- Sa syntaxe est la suivante:

fscanf(nom_flot, "*chaîne de contrôle*" , argument1,..., argumentN);

```
#include <stdio.h>
FILE *fp; char ch1[10]; char ch2[10];
if ((fp = fopen("donnees","a+")) == NULL) {
fprintf(stderr,"Impossible d'ouvrir le fichier données en
lecture\n");}
fscanf( fp, "%s %s",ch1,ch2);
printf("%s %s ",ch1,ch2);
```

Lecture et écriture de caractère

- Similaires aux fonctions getchar et putchar, les fonctions fgetc et fputc permettent respectivement de lire et d'écrire un caractère dans un fichier.
- **Prototypes:**
  - int fgetc(FILE* *fp*);
  - int fputc(int *caractere*, FILE **fp*);
- La fonction **fgetc()** retourne le code ASCII du caractère lu à partir du fichier, ou **EOF** en cas de lecture de la fin de fichier ou en cas d'erreur.
- La fonction **fputc()** retourne le code ASCII du caractère écrit dans le fichier, ou **EOF** en cas d'erreur.

Programme qui lit le contenu d'un fichier texte, et le recopie caractère par caractère dans un autre fichier:

```
#include <stdio.h>
#define ENTREE "c:\\algo\\essai.txt"
#define SORTIE "c:\\algo\\essai2.txt"
int main(){
FILE *fp_in, *fp_out;       int c;
if ((fp_in = fopen(ENTREE,"r")) == NULL){
 printf("\nErreur: Impossible de lire dans le fichier %s\n",ENTREE);   return 0;  }
if ((fp_out = fopen(SORTIE,"w")) == NULL){
 printf("\nErreur: Impossible d'écrire dans le fichier %s\n", SORTIE);  return 0;  }
c = fgetc(fp_in);
while (c != EOF)      { //tant qu'on n'a pas lu la fin du fichier
        fputc(c, fp_out);
        c = fgetc(fp_in)     }
fclose(fp_in);
fclose(fp_out);
return 1;}
```

Lecture et écriture de chaîne

- La fonction **fgets** permet la lecture d'une chaîne de caractères ch à partir d'un flot. Son prototype est:
- fgets (char * chaîne, int n, FILE * fp)
  - *chaîne* est de type pointeur vers char et doit pointer vers un tableau de caractères.
  - *taille* est la taille en octets du tableau de caractères pointé par *chaîne*.
  - *flot-de-données* est de type pointeur vers FILE. Il pointe vers le fichier à partir duquel se fait la lecture.
- La fonction fgets rend le pointeur *chaîne* cas de lecture sans erreur, ou NULL dans le cas de fin de fichier ou d'erreur.
- La lecture s'effectue jusqu'à concurrence de **n**-1 caractères, ou lorsque le caractère **n** est lu

Lecture et écriture de chaîne

- Exemple d’ Utilisation typique

```
#include <stdio.h>
#define LONG ...
char ligne[LONG];
FILE *fi;
while (fgets(ligne,LONG,fi) != NULL) /* stop sur fin de
fichier ou erreur */
 {
 ...
/* utilisation de ligne */
 }
```

Lecture et écriture de chaîne

- La fonction **fputs** permet l'écriture d'une chaîne de caractères ch dans un fichier. Son prototype est:
- int fputs(const char * chaîne, FILE * stream);
  - *chaîne* est de type pointeur vers char. Pointe vers un tableau de caractères contenant une chaîne se terminant par un *null*.
  - *stream* est de type pointeur vers FILE. Il pointe vers le fichier sur lequel se fait l'écriture.
- **fputs** retourne le code ASCII du dernier caractère écrit, ou **EOF** en cas d'erreur.

```
char ch[]= "Hello world";
FILE * fp;
...
fputs( ch, fp);
```

Remarques

- Toutes les fonctions d'écriture qu'on vient de voir (fprintf, fputc, fputs) peuvent être utilisées pour écrire sur la sortie standard (l'écran). Dans ce cas, le flot est **stdout**.
- Toutes les fonctions de lecture qu'on vient de voir (fscanf, fgetc, fgets) peuvent être utilisées pour lire à partir de l'entrée standard (le clavier). Dans ce cas, le flot est **stdin**.

Lecture et écriture dans un fichier binaire

- Pour un fichier ouvert en mode binaire, les fonctions de lecture et d'écriture à utiliser sont respectivement **fread** et **fwrite**. Ces fonctions permettent de transférer des blocs d'octets. La syntaxe de fread est:
  - **fread(adresse,taille_bloc, nb_bloc, nom_flot);**
- Lit à partir du fichier un nombre **nb_bloc** de blocs dont chacun est de taille **taille_bloc** et les copie en mémoire à partir de l'adresse **adresse**
- **fread** retourne le nombre de blocs lus si aucune erreur n'est intervenue. Ce nombre peut être inférieur au nombre **nb_bloc** si la fin de fichier a été rencontrée ou lorsqu'une erreur intervient.

Lecture et écriture dans un fichier binaire

- La syntaxe de fwrite est:
  - **fwrite(adresse,taille_bloc, nb_bloc, nom_flot);**
- Écrit dans le fichier un nombre **nb_bloc** de blocs dont chacun est de taille **taille_bloc** lus en mémoire à partir de l'adresse **adresse**.
- **fwrite** retourne le nombre de blocs écrits. Une erreur a eu lieu si ce nombre est inférieur à **n**.

```
int tab[ ] = {2,8,6,-9,7,6,78,25,0,6};
FILE * fp;
...
fwrite(tab, sizeof(int), 10, fp); /* fwrite(tab, 10*sizeof(int), 1,
fp); */
```

La fonction fseek

- La fonction **fseek** permet déplacer le curseur à un endroit précis du fichier. Sa syntaxe est:
  - **fseek(nom_flot, *deplacement*, *origine*);**
- L'argument *deplacement* représente le déplacement (positif ou négatif) en nombre d'octets à effectuer dans le fichier. Il s'agit d'un déplacement relatif par rapport à l'argument *origine.* Il doit être de type **long.**
- L'argument *origine* peut prendre trois valeurs:
  - SEEK_SET (ou 0): début du fichier;
  - SEEK_CUR (ou 1): position courante;
  - SEEK_END (ou 2): fin du fichier.

La fonction rewind

- La fonction **rewind** permet de déplacer le curseur au début du fichier. Sa syntaxe est:
  - **rewind(nom_flot);**
- Elle est équivalente à:
  - **fseek(*flot*, 0, SEEK_SET);**

La fonction feof

- On peut combiner les fonctions de lecture avec la fonction **feof** pour vérifier si on a atteint la fin d'un fichier. Cette fonction a pour prototype:
- int feof(FILE *fp);

```
FILE * fp;
...
ouverture du fichier vérification du succès de l'ouverture du fichier
...
lecture sur le fichier
while(!feof(fp))  {
traitement des données lues
lecture sur le fichier
}
...
```

FIN