

Programmation avancée

Les Fichiers en C

Walter Rudametkin

Walter.Rudametkin@polytech-lille.fr
<https://rudametw.github.io/teaching/>

Bureau F011
Polytech'Lille

23 février 2016

1 / 18

Les fichiers en C

Pas de fichiers dans le langage C

- ▶ Bibliothèque standard `libc.so/libc.a` : inclure le fichier entête `<stdio.h>`

Fichier C = une suite d'octets (= flot)

- ▶ Pas de fichiers typés

Fichiers texte

- ▶ Les octets représentent des caractères codant les données (souvent le très limité ASCII, mais aussi le populaire et recommandé **UTF-8**)
- ▶ Standard – Éditables – Imprimables

2/18

Fichiers Binaires

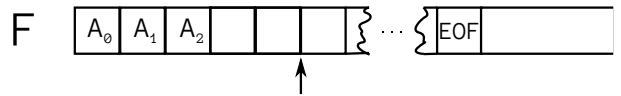
- ▶ Octets représentent la copie exacte des données en mémoire sur un système donné
- ▶ Non standard – Non éditables – Non imprimables
- ▶ Mais lecture / écriture plus rapides (pas d'analyse)
- ▶ En général, plus compacts
 - ▶ ex: 654875 = 6 octets (char), 2/4 octets (short/int)

Pas d'attribut *texte* ou *binaire* sur un fichier
(dépend de l'interprétation des octets)

- ▶ N'intervient pas à la déclaration
- ▶ Lié aux opérations applicables

3/18

Le type FILE



- ▶ Défini dans `<stdio.h>`
- ▶ Structure C contenant
 - ▶ Identification du fichier associé (descripteur)
 - ▶ Position du curseur dans le fichier
 - ▶ Tampon de lecture / écriture
 - ▶ Indication de mode d'ouverture ...
- ▶ Opérations sont effectuées sur un `FILE` * fourni à l'ouverture

4/18

Fichiers texte: ouverture

- ▶ Défini dans `<stdio.h>`
- ▶ `FILE * fopen(char *nom, char *mode)` où

$$\text{mode} = \begin{cases} \text{r} : & \text{lecture} \\ \text{w} : & \text{création/écriture} \\ \text{a} : & \text{allongement} \end{cases}$$

lecture: F

A_0	A_1	A_2				...	EOF
-------	-------	-------	--	--	--	-----	-----

écriture: F

allongement : F

A ₀	A ₁	A ₂		⋮	A _{n-1}	A _n	EOF
----------------	----------------	----------------	--	---	------------------	----------------	-----

↑

5/18

Fichiers texte: ouverture/fermeture

[Retour](#)

- ▶ FILE * si tout va bien
- ▶ NULL si erreur (ex: fichier inexistant, pas les droits de lecture ou écriture, ...)

Fermeture

- ▶ `fclose(FILE *fp)`
- ▶ Déconnecte fp du fichier physique

6/18

Fichiers texte: exemple

```

1  #include <stdio.h>
2
3  int main(){
4      FILE *fp = fopen("toto.txt", "r");
5      if (fp == NULL) printf("toto.txt inaccessible: fp=%p\n", fp);
6      else printf("toto.txt accessible: fp=%p\n", fp);
7
8      if (fp != NULL){
9          printf("Fermer toto\n");
10         fclose(fp);
11     }
12 }

```

si toto.txt existe

toto.txt accessible: fp=0x1d12010
Fermer toto

si toto.txt n'existe pas

toto.txt inaccessible: fp=(nil)

7/18

Fichiers texte: utilisation

- Généralisation des manipulations effectuées sur l'entrée/sortie standard (stdin, stdout)
- Dans <stdio.h>
 - entrée standard : FILE * stdin
 - sortie standard : FILE * stdout
- Connexion à l'exécution aux entrées / sorties standard fournies par le système (console par défaut, redirigeables par < >)
- Lectures écritures à partir de la position suivant le curseur

8/18

Fichiers texte: lecture

- char getc (FILE *fp)
 - getchar() ⇔ getc(stdin)
- int fscanf(FILE *fp, char *format, ...)
- scanf(...) ⇔ fscanf(stdin, ...)
- Retourne le nombre d'items lus
- char * fgets(char *chaine, int taille, FILE *fp)
- int feof(FILE *fp)
 - Retourne une valeur différent à zéro si la fin du fichier a été rencontrée lors d'une opération de lecture (valeur lue indéterminée)

9/18

Fichiers texte: mode écriture/allongement

- int putc(char c, FILE *fp)
 - putchar(c) ⇔ putc(c, stdout)
- int fprintf(FILE *fp, char *format, ...)
- int fputs(char *chaine, FILE *fp)

10/18

Fichiers texte: exemple

```

1  #include <stdio.h>
2
3  int main() {
4      FILE *fp1, *fp2;
5      if ((fp1 = fopen("titi.txt", "r")) != NULL) {
6          if ((fp2 = fopen("titibis.txt", "w")) != NULL) {
7              int c = getc(fp1);
8              while (!feof(fp1)){
9                  putc(c, fp2);
10                 c = getc(fp1);
11             }
12         }
13     }
14     if (fp1 != NULL) fclose(fp1);
15     if (fp2 != NULL) fclose(fp2);
16 }

```

11/18

Fichiers binaires: ouverture

- Octets représentent la copie exacte du codage des données en mémoire

Ouverture

- FILE *fopen (char *nom, char *mode) où

mode = {
 rb : lecture
 wb : création/écriture
 ab : allongement
 rb+ : lecture/écriture
 ...
}

lecture : F [A₀ A₁ A₂ ...] {EOF}

écriture : F []

lecture/écriture : F [A₀ A₁ A₂ ...] {EOF}

12/18

Fichiers binaires: fermeture/écriture

Fermeture (idem fichiers texte)

- ▶ `fclose(FILE *fp)`
- ▶ `int feof(FILE *fp)`

Écriture (mode création ou lecture/écriture)

- ▶ `int fwrite(void *pt, int taille, int nb, FILE *fp)`
Écrit sur le fichier `fp`, à partir de la position suivant le curseur, `nb` objets, chacun de taille `taille`, qui se trouvent contiguëment dans la zone mémoire pointée par `pt`.
- ▶ Utilisation courante :
`FILE *fp; <T> x;`
`fwrite(&x, sizeof(x), 1, fp);`

13/18

Fichiers binaires: écriture

Mode lecture/écriture

- ▶ `char x = '??';`
`fwrite(&x, 1, 1, fp);`

14/18

Fichiers binaires: lecture

- ▶ `int fread(void *pt, int taille, int nb, FILE *fp)`
Lire `nb` objets de taille `<taille>` et les copier dans l'espace pointé par `pt`
- ▶ Utilisation courante :
`<T> x;`
`fread(&x, sizeof(<T>), 1, fp);`

15/18

Fichiers binaire: lecture

```
1  #include <stdio.h>
2
3  typedef struct{
4      char nom[30];
5      int age;
6  } Personne;
7
8  int main(){
9      Personne P;
10     FILE * fich = fopen("personnes" , "rb");
11     if(fich!=NULL){
12         fread(&P, sizeof(Personne), 1, fich);
13         while (!feof(fich)) {
14             printf("%s %d\n", P.nom, P.age);
15             fread(&P, sizeof(Personne), 1, fich);
16         }
17         fclose(fich);
18     }
19 }
```

16/18

Fichiers binaires: accès direct

- ▶ `int fseek(FILE *fp, long déplacement, int origine)`

où $origine = \begin{cases} SEEK_SET: & \text{début} \\ SEEK_CUR: & \text{position courant} \\ SEEK_END: & \text{fin} \end{cases}$
- ▶ Positionne le curseur pour la prochaine lecture ou écriture
- ▶ Position = déplacement + origine
- ▶ Usage courant :
`fseek(fp, i*sizeof(<T>), SEEK_SET);`

17/18

Fichiers: conclusion

- ▶ *Texte* ou *binaire* n'est pas un attribut de fichier
- ▶ Un fichier texte peut être exploité en binaire comme simple suite d'octets
 - ▶ ex : `fseek(fp, i*sizeof(char), SEEK_SET);`
 - ▶ ex : utilisation de `fread` ou `fwrite` sur un fichier texte
- ▶ Exploitation d'un fichier binaire en texte ?????

18/18