# Programmation avancée Les Fichiers en C

#### Walter Rudametkin

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

> Bureau F011 Polytech'Lille

> > CM5

#### 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

0/10

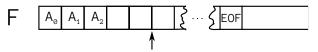
# Fichiers Binaires

- 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

## 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

4/1

## Fichiers texte: ouverture

- ► Défini dans <stdio.h>
- ► FILE \* fopen(char \*nom, char \*mode) Où

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

# 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

6/1

# Fichiers texte: exemple

```
#include <stdio.h>

int main(){
    FILE *fp = fopen("toto.txt", "r");
    if (fp == NULL) printf("toto.txt inaccessible: fp=%p\n",fp);
    else printf("toto.txt accessible: fp=%p\n",fp);

if (fp != NULL){
    printf("Fermer toto\n");
    fclose(fp);
}

sitoto.txt existe
toto.txt accessible:fp=0x1d12010
Fermer toto
    fp=(nil)
```

#### 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

1/10

#### Fichiers texte: lecture

- ▶ int fscanf(FILE \*fp, char \*format, ...)
  - $scanf(...) \Leftrightarrow fscanf(stdin, ...)$
  - ▶ Retourne le nombre d'items lus
- ▶ 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)

# 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

#### Fichiers binaires: ouverture

 Octets représentent la copie exacte du codage des données en mémoire

#### Ouverture

► FILE \*fopen (char \*nom, char \*mode) Où

```
\mbox{mode} = \begin{cases} \mbox{rb}: & \mbox{lecture} \\ \mbox{wb}: & \mbox{création/écriture} \\ \mbox{ab}: & \mbox{allongement} \\ \mbox{rb+}: & \mbox{lecture/écriture} \end{cases}
```

12/1

11/18

#### Fichiers binaires: fermeture/écriture

Fermeture (idem fichiers texte)

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

#### Écriture (mode création ou lecture/écriture)

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

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

1/10

## Fichiers binaires: lecture

int fread(void \*pt, int taille, int nb,
FILE \*fp)
Lire nb objets de taille <taille> et les copier dans

Utilisation courante :

l'espace pointé par pt

```
<T> x;
fread(&x, sizeof(<T>), 1, fp);
```

Fichiers binaire: lecture

```
tinclude (stdio.h)

typedef struct{
    char nom[30];
    int age;
} Personne;

int main(){
    Personne P;
    if(fich!=NULL){
        fread(&P, sizeof(Personne), 1, fich);
        while (!feof(fich)) {
            printf("%s %d\n", P.nom, P.age);
            fread(&P, sizeof(Personne), 1, fich);
            fread(&P, sizeof(Personne), 1, fich);
}

fread(&P, sizeof(Personne), 1, fich);
}

fclose(fich);
}
```

15/18

#### Fichiers binaires: accès direct

```
\label{eq:output} \begin{array}{ll} \text{où} & \text{origine} = \begin{cases} \text{SEEK\_SET:} & \text{d\'ebut} \\ \text{SEEK\_CUR:} & \text{position courant} \\ \text{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);
```

#### 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 ?????

17/18

18/18