



# Chap. 10

## Fichiers binaires

# I2181A

## Langage C : modularisation

Anthony Legrand

- Une suite (*flot* ou *flux*) de bytes
- On parle souvent de *streams* (un *stream* peut être autre chose qu'un fichier ; ex: socket)
- Souvent accédé séquentiellement

## 2 types de fichiers

- Fichiers **textes** qui contiennent des caractères
- Fichiers **binaires** qui contiennent autre chose (image, film, db...)

# Descripteur de fichier

- En C, les informations utiles relatives aux fichiers sont stockées dans une **structure FILE**
- Ces structures sont stockées dans un tableau. On y accède:
  - soit par un indice (*file descriptor*)
  - soit à l'aide d'un pointeur (**FILE\***)

# 3 étapes

- 1) Ouvrir un fichier
- 2) Lire ou écrire dans le fichier
- 3) Fermer le fichier

# Ouvrir un fichier binaire

```
#include <stdio.h>
```

```
FILE* fopen (char* path, char* mode)
```

- **path**: chemin du fichier
- **mode**:
  - **"rb"** → lire en binaire
  - **"wb"** → écrire en binaire
  - **"ab"** → étendre en binaire

# Ouvrir un fichier (résultat)

- En cas de réussite, on a **un pointeur sur une structure FILE**
- Sinon, on reçoit **NULL** et ***errno* est positionné**

```
FILE* fin = fopen(nom, "wb");  
if (fin == NULL) {  
    perror("Problème"); // message lié à errno  
    exit(1);  
}
```

# Ecrire dans un fichier binaire

8

```
size_T fwrite (void* base, size_t size,  
               size_t nmemb, FILE* stream)
```

- **base**: une zone mémoire de taille  $\geq (\textit{size} * \textit{nmemb})$
- Ecrit **nmemb** enregistrements de **base** (de taille **size** bytes) dans le fichier **stream**
- Renvoie le nombre d'enregistrements effectivement écrits dans stream



# Ecrire dans un fichier (résultat)

9

Si le nombre d'enregistrements écrits  
est **< nmemb** → problème d'écriture

# Ecrire dans un fichier: exemple

10

```
FILE* f; // fichier binaire ouvert en écriture
struct Point data[NB];
...
if (fwrite(data, sizeof(struct Point), NB, f) != NB)
{
    // erreur d'écriture
}
```

# Ecrire une chaîne de caractères dans un fichier: ex

```
FILE* f; // fichier binaire ouvert en écriture
char *s = "HELLO WORLD";
size_t nbr = strlen(s) + 1
if (fwrite(s,sizeof(char),nbr,f) != nbr) {
    // erreur d'écriture
}
```

Ne pas écrire d'adresse  
dans un fichier!



# Lire dans un fichier binaire

12

```
size_T fread (void* base, size_t size,  
              size_t nmemb, FILE* stream)
```

- **base**: une zone mémoire de taille  $\geq (\textit{size} * \textit{nmemb})$
- Lit au plus **nmemb** enregistrements de taille **size** bytes dans le fichier **stream** et les place en mémoire à l'adresse **base**
- Renvoie le nombre d'enregistrements effectivement lus dans stream

# Lire dans un fichier (résultat)

13

Si le nombre d'enregistrements lus est **< nmemb** → fin du fichier atteinte ou problème de lecture, à vérifier grâce à :

- la fonction **ferror** (cf man)

ou

- la fonction **feof** (cf man)

# Lire dans un fichier: exemple

14

```
FILE* f; // fichier binaire contenant
        // des Point, ouvert en lecture
struct Point tmp[NB];
size_t nread = NB;
while (nread == NB) {
    nread = fread(tmp, sizeof(struct Point), NB, f);
    // traitement des nread points lus
}
if (ferror(f)) { // OU if (!feof(f))
    // erreur de lecture
}
```

# Fermer un fichier

On ferme le fichier pour libérer les ressources liées à ce fichier

```
#include <stdio.h>
```

```
int fclose (FILE *fp)
```

- En cas de réussite, on reçoit **0**
- Sinon, on reçoit **EOF** et **errno** est positionné

# Fonctions supplémentaires

16

- **fflush**: vide le buffer d'écriture (càd. écriture physique sur le disque)
- **fseek**, **ftell**, **rewind**: gestion des lectures non-séquentielles
- **rename**: renomme un fichier
- **remove**: supprime un fichier
- **perror**: affiche sur **stderr** un message d'erreur système (lié à **errno**), en plus du message passé en paramètre