

Programmation C

Fichiers

ING1

CY Tech

Introduction

Introduction

Fichier

- Programme : traitement sur des données
- Exige la disponibilité des données en mémoire
- Peut exiger d'avoir une "persistance" des données : utilisation des fichiers
- Fichier : mémoire de masse, stocké sur un support "permanent"

Introduction

Fichier au niveau système

- Emplacement "disque"
- Droits
- États (ouverture, pointeur de fichier, positionnement, ...)
- Accès séquentiel : besoin de "parcourir" pour accéder à une donnée

Introduction

Fichier binaire/ASCII

- Traditionnellement distinction : fichier binaire / fichier ASCII
- Fichier ASCII : fichier stockant des caractères
- Fichier binaire : fichier stockant des données binaires
- Réaction différente selon les systèmes (utilisation)
- Unix : pas ou peu d'importance (un seul type de fichier)
- Réaction différente selon les langages (structuration)
- C : pas de structuration à l'avance

Introduction

Type d'accès

- Bas niveau : dépendant du système (open, close, ...)
 - ▶ Manipulation des file descriptor (fd)
- Haut niveau : indépendant du système (fopen, fclose, ...)
 - ▶ Manipulation d'une structure : FILE

Listing 1 – Structure FILE pré-défini

```
struct FILE {  
    char* file;  
    char* ptr;  
    int cnt;  
    int flags;  
    int fd;  
}
```

Introduction

Gestion des entrées sorties

- Utilisation des consoles POSIX¹
 - ▶ *fd* : FILE*
 - ▶ 0 : stdin
 - ▶ 1 : stdout
 - ▶ 2 : stderr
- Utilisation de la bibliothèque stdio
- Fonctions :
 - ▶ scanf
 - ▶ printf
 - ▶ getchar, ...
 - ▶ Travaillent sur les flux adéquats

Introduction

Utilisation des fichiers

- Redirection : permet d'utiliser des fichiers
 - ▶ Fichier d'entrée : redirection de `stdin`
 - ▶ Fichier de sortie : redirection de `stdout`
 - ▶ Fichier de sortie d'erreur : redirection de `stderr`
- Se fait via la ligne de commande
- Peut entraîner un mauvais affichage

Introduction

Modes d'ouverture

- Plusieurs possibilités d'ouverture
 - ▶ Non destructive : lecture
 - ▶ Potentiellement destructive : écriture, ajout
- Possibilité d'échec
 - ▶ Problème de droits
 - ▶ Évolution possible du système de fichier (modification, suppression du fichier par un autre programme, ...)

Manipulation bas niveau

Manipulation bas niveau

Primitives d'ouverture

- Dépendant du système : peu portable
- Nécessite beaucoup d'inclusion²
- `int open(const char* pathname, int flags)`
 - ▶ Retour : *file descriptor*
 - ▶ `pathname` : fichier (avec le chemin d'accès)
 - ▶ `flags` : mode d'ouverture

Manipulation bas niveau

Modes d'ouverture

- Manipulation de constantes symboliques
- Les plus courantes :
 - ▶ O_APPEND : ouverture en ajout (marqueur positionné en fin de fichier)
 - ▶ O_CREAT : ouverture en création (possibilité d'écrasement)
 - ▶ O_EXCL : pas d'écrasement de fichier si existant
 - ▶ O_RDWR : ouverture en lecture/écriture (positionnement en début)
- Droits d'accès :
 - ▶ S_IRWXU : 700
 - ▶ S_IRUSR : 400
 - ▶ S_IWUSR : 200
 - ▶ S_IXUSR : 100
 - ▶ Existence de macros pour les autres catégories (*group, other*)

Manipulation bas niveau

Primitives d'écriture / lecture

- Prototypes :
 - ▶ `ssize_t read (int fd, void *buf, size_t count)`
 - ▶ `ssize_t write (int fd, const void *buf, size_t count)`
- `read` : lecture depuis un fichier
- `write` : écriture depuis un fichier
- Retour : nombre d'octets lus ou écrits
- Paramètres :
 - ▶ `fd` : *file descriptor*
 - ▶ `buf` : donnée
 - ▶ `count` : taille

Manipulation bas niveau

Primitives de fermeture

- Prototype : `int close (int fd)`
- Permet de fermer un fichier
- Tout fichier ouvert doit être fermé
 - ▶ Permet de forcer la fin des transitions
 - ▶ Permet d'assurer la cohérence du système de fichier
- Erreurs possibles :
 - ▶ Mauvais *file descriptor*
 - ▶ Interruption par signaux
 - ▶ Problème d'entrée/sortie

Manipulation bas niveau

D'autres possibilités

- Manipulation directe du système de fichiers
- `rename` : changement de nom
- `chmod` : modification des droits
- `chown` : modification de l'appartenance
- `stat` : récupération d'informations sur un fichier
- `access` : test l'accès à un fichier
- `symlink` : création de liens symboliques
- ...

Manipulation bas niveau

Exemple

- cf `fichier.c`

Manipulation haut niveau

Manipulation haut niveau

Introduction

- Permet de s'abstraire du système de fichier : normalement "portable"
- Manipulation de flux
- Nom des fonctions : $f\alpha \Leftrightarrow fopen, fread, fwrite$
- Même comportement que les fonctions de bas niveau
- Fonctions définies dans `stdio`

Manipulation haut niveau

Nouvelles fonctions

- `ferror` : état du flux
- `feof` : teste la fin d'un flux
- `fseek`, `fsetpos` : repositionne un flux
- `ftell`, `fgetpos` : récupère la position d'un flux
- `setbuf` : opérations sur les buffers

Manipulation haut niveau

Nouvelles fonctions

- `fgetc` : lecture du caractère suivant (ne pas utiliser `getc`) d'un flux
- `fgets` : lecture d'une chaîne de caractères d'un flux
- `fprintf` : sortie formatée sur un flux
- `fscanf` : entrée formatée depuis un flux
- `fputc` : met un caractère sur un flux
- `fputs` : met une chaîne de caractères sur un flux
- `mktemp` : crée un nom de fichier temporaire

Manipulation haut niveau

Exemple

- cf `ffichier.c`