Méthodologie de la programmation *–E2I.1-*

Les fichiers

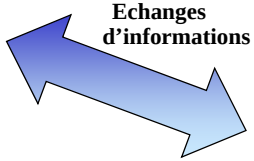
Cyrille CHAVET

Vue d'ensemble

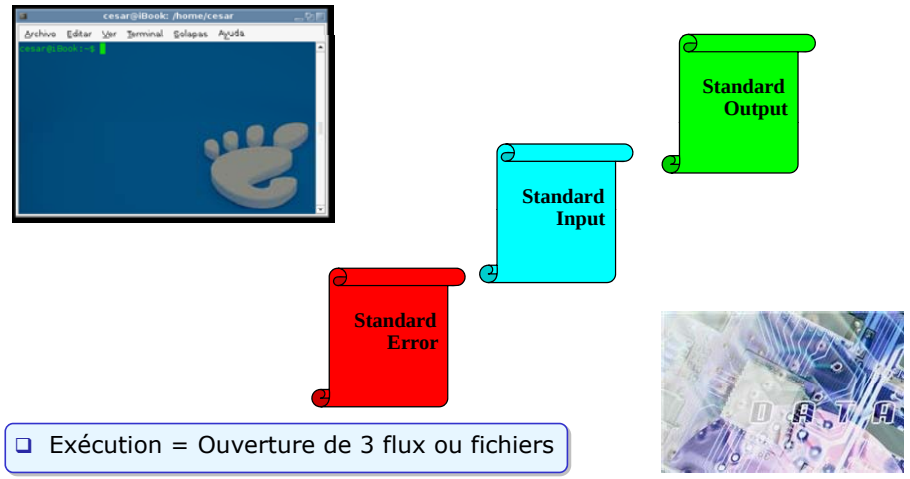
- Notion de flux
- Ouverture d'un fichier
- Modification d'un fichier

Plan

Notion de flux

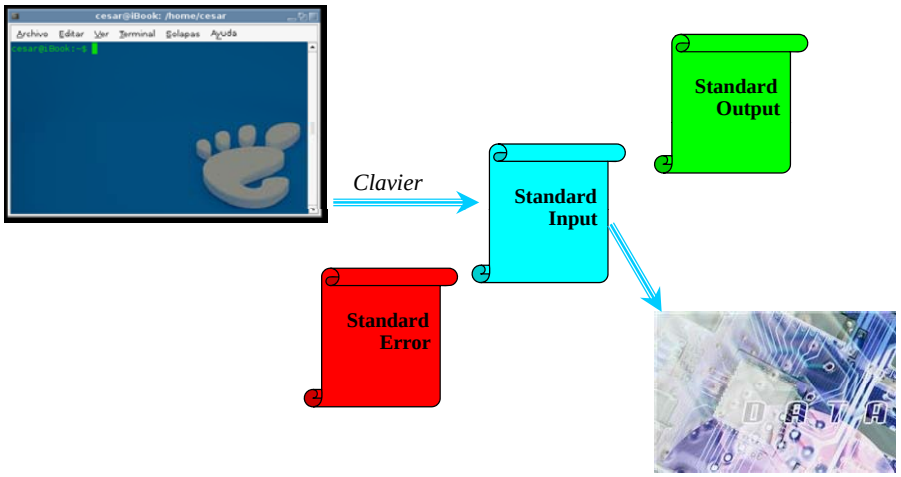


Flux standards

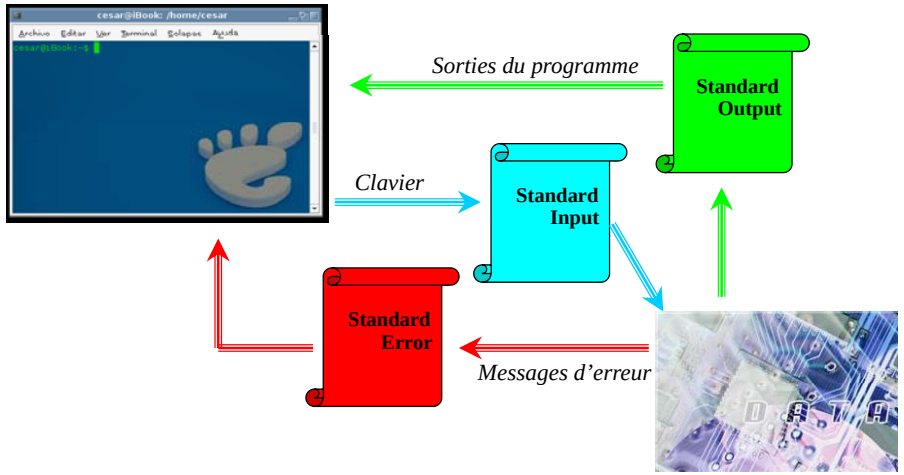


- Exécution = Ouverture de 3 flux ou fichiers

Flux standards



Flux standards



Notion de fichier

- Trois constantes prédéfinies dans *stdio.h* :
  - *stdin*, *stdout* et *stderr*
  - Type pointeur vers *FILE*

Notion de fichier

- Trois constantes prédéfinies dans *stdio.h* :
  - *stdin*, *stdout* et *stderr*
  - Type pointeur vers *FILE*

- Un type « fichier » : *FILE*
- Fichier = ensemble d'octets stocké dans une mémoire
- Deux classes de fichiers:
  - Texte : suite de caractères regroupés en lignes
  - Binaire : composé d'une suite d'octets
- Un fichier est identifié
  - À l'extérieur d'un programme par un ***nom physique*** qu'il lui est attribué à sa création
  - À l'intérieur d'un programme par un ***pointeur*** vers une structure *FILE*

Plan

Ouverture / Fermeture d'un fichier

- Faire le lien entre le *nom physique* et le *pointeur* sur le fichier

Ouverture / Fermeture d'un fichier

- Faire le lien entre le *nom physique* et le *pointeur* sur le fichier

```
#include <stdio.h>

FILE *fp;
```

fp: *Pointeur sur un élément de type FILE*

Ouverture / Fermeture d'un fichier

- Faire le lien entre le *nom physique* et le *pointeur* sur le fichier

```
#include <stdio.h>

FILE *fp;
```

fp: *Pointeur sur un élément de type FILE*

- Le fichier n'existe pas encore !

Ouverture / Fermeture d'un fichier

- Faire le lien entre le *nom physique* et le *pointeur* sur le fichier

```
#include <stdio.h>

FILE *fp;
```

fp*: Pointeur sur un élément de type FILE*

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

nom: nom physique du fichier
mode: chaîne de caractères précisant la manière dont on veut utiliser le fichier

Ouverture / Fermeture d'un fichier

- Faire le lien entre le *nom physique* et le *pointeur* sur le fichier

```
#include <stdio.h>

FILE *fp;

…/…

fp = fopen("MonFichier", "w");
```

fp*: Pointeur sur un élément de type FILE*

- Ouverture réussie: *fp* pointe sur le fichier
- Echec: *fp == NULL*

Ouverture / Fermeture d'un fichier

- Faire le lien entre le *nom physique* et le *pointeur* sur le fichier

```
#include <stdio.h>

FILE *fp;

…/…

fp = fopen("MonFichier", "w");
```

fp*: Pointeur sur un élément de type FILE*

fopen*: Ouverture / Création du fichier*

Ouverture / Fermeture d'un fichier

- Faire le lien entre le *nom physique* et le *pointeur* sur le fichier

```
#include <stdio.h>
FILE *fp;
…/…
fp = fopen("MonFichier", "w");
…/…
fclose(fp);
```

fp*: Pointeur sur un élément de type FILE*

fopen*: Ouverture / Création du fichier*

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


Ouverture / Fermeture d'un fichier

- Faire le lien entre le *nom physique* et le *pointeur* sur le fichier

```
#include <stdio.h>
FILE *fp;
…/…
fp = fopen("MonFichier", "w");
…/…
fclose(fp);
```

fp*: Pointeur sur un élément de type FILE*

fopen*: Ouverture / Création du fichier*

fclose*: Fermeture du fichier*

Ouverture / Fermeture d'un fichier

- Fin de fichier: *EOF*
  - Constante dans *stdio.h*
- Trouver la fin d'un fichier
  - *int feof(FILE *fichier)*

Paramètres d'ouverture

- Fichier Texte

Mode	Lecture	Ecriture	Remarque
r	√	-	Le fichier doit exister
w	-	√	Ouverture ou création du fichier
a	-	√	Ouverture ou création du fichier, Ecriture en fin de fichier

Paramètres d'ouverture

- Fichier Texte

Mode	Lecture	Ecriture	Remarque
r	√	-	Le fichier doit exister
w	-	√	Ouverture ou création du fichier
a	-	√	Ouverture ou création du fichier, Ecriture en fin de fichier
r+	√	√	Le fichier doit exister
w+	√	√	Ouverture ou création du fichier
a+	√	√	Ouverture ou création du fichier, Ecriture en fin de fichier

Paramètres d'ouverture

- Fichier Binaire

Mode	Lecture	Ecriture	Remarque
rb	√	-	Le fichier doit exister
wb	-	√	Ouverture ou création du fichier
ab	-	√	Ouverture ou création du fichier, Ecriture en fin de fichier
rb+	√	√	Le fichier doit exister
wb+	√	√	Ouverture ou création du fichier
ab+	√	√	Ouverture ou création du fichier, Ecriture en fin de fichier

Exemple

- Ouverture en lecture et test de validité

```
#include <stdio.h>

FILE *fp;

if ((fp = fopen("Donnees","r")) == NULL)
{
        fprintf(stderr,"Impossible d'ouvrir le fichier données en lecture\n");
        exit(1);
}
```


Plan

Fonctions d'écriture/lecture classiques

- Lecture/écriture formattées
  - Pour lire : **fscanf**
    - *int fscanf(FILE*, char *, param$_1$, …);*
  - Pour écrire : **fprintf**
    - *int fprintf(FILE*, char *, param$_1$,…);*

Fonctions d'écriture/lecture classiques

- Lecture/écriture formattées
  - Pour lire : **fscanf**
    - *int fscanf(FILE*, char *, param$_1$, …);*
  - Pour écrire : **fprintf**
    - *int fprintf(FILE*, char *, param$_1$,…);*

```
Int NbreSlidesRestant = 5;
…/…
fprintf(fp, ″Ce cours se termine dans %d slides\n″ , NbreSlidesRestant);
…/…
```


Fonctions d'écriture/lecture classiques

- Lecture/écriture formattées
  - Pour lire : **fscanf**
    - *int fscanf(FILE*, char *, param$_1$, …);*
  - Pour écrire : **fprintf**
    - *int fprintf(FILE*, char *, param$_1$,…);*

```
Int NbreSlidesRestant = 5;
…/…
fprintf(fp, ″Ce cours se termine dans %d slides\n″ , NbreSlidesRestant);
…/…
```

Fichier pointé par fp

Ce cours se termine dans 5 slides

Fonctions d'écriture/lecture classiques

- Lecture/écriture par caractères
  - Pour lire : **fgetc**
    - *int fgetc(FILE*);*
  - Pour écrire : **fputc**
    - *int fputc(int, FILE*);*

Fonctions d'écriture/lecture classiques

- Lecture/écriture par caractères
  - Pour lire : **fgetc**
    - *int fgetc(FILE*);*
  - Pour écrire un entier : **fputc**
    - *int fputc(int, FILE*);*

```
#include <stdio.h>

int c;
FILE *fp;

…/…

while ((c = fgetc(fp)) != EOF)
{
        ...          /* utilisation de c */
}
```


Fonctions d'écriture/lecture classiques

- Lecture/écriture par ligne
  - Pour lire : **fgets**
    - *char* fgets(char*, int, FILE*);*
    - Valeur entière: taille en octets du tableau de caractères pointé par *char**
  - Pour écrire une chaîne : **fputs**
    - *int fputs(char*, FILE*);*

Fonctions d'écriture/lecture classiques

- Lecture/écriture par ligne
  - Pour lire : **fgets**
    - *char* fgets(char*, int, FILE*);*
    - Valeur entière: taille en octets du tableau de caractères pointé par *char**
  - Pour écrire : **fputs**
    - *int fputs(char*, FILE*);*

- Sur fin de fichier ou erreur, fgets rend NULL et non pas EOF.

Grrr …

Fonctions d'écriture/lecture classiques

- Lecture/écriture par ligne
  - Pour lire : **fgets**
    - *char* fgets(char*, int, FILE*);*
    - Valeur entière: taille en octets du tableau de caractères pointé par *char**
  - Pour écrire : **fputs**
    - *int fputs(char*, FILE*);*

```
#include <stdio.h>
#define LONG 10
char ligne[LONG];
FILE *fp;

while (fgets(ligne, LONG, fp) != NULL)      /* stop sur fin de fichier ou erreur*/
{
        ... /* utilisation de ligne */
}
```

Fonctions exotiques

- Lecture
  - Fonction : **fread**
    - *size_t fread(void** cible, *size_t* lataille, *size_t* nombre, *FILE** fichier*);*
    - Lit **nombre** objets dans **fichier**, chacun de taille **lataille**, et les copie les uns à la suite des autres dans l'espace pointé par cible.
    - Retourne le nombre d'objets effectivement lus, qui peut être inférieur au nombre demandé, à cause de la rencontre de la fin du fichier *EOF*, d'une erreur, etc.

Fonctions exotiques

- Lecture
  - Fonction : **fread**
    - *size_t fread(void** cible, *size_t* lataille, *size_t* nombre, *FILE** fichier*);*
    - Lit **nombre** objets dans **fichier**, chacun de taille **lataille**, et les copie les uns à la suite des autres dans l'espace pointé par cible.
    - Retourne le nombre d'objets effectivement lus, qui peut être inférieur au nombre demandé, à cause de la rencontre de la fin du fichier *EOF*, d'une erreur, etc.
- Ecriture
  - Fonction : **fwrite**
    - *size_t fwrite(void** source, *size_t* lataille, *size_t* nombre, *FILE** fichier*);*
    - Ecrit les **nombre** objets, de taille **lataille**, qui se trouvent les uns à la suite des autres à l'adresse indiquée par **source**.
    - Retourne le nombre d'objets écrits, qui peut être inférieur au nombre demandé (en cas d'erreur).

Fonctions exotiques

- Positionnement
  - Fonction : **fseek**
    - *int fseek(FILE *fp, long deplacement, int origine)*
    - Positionne le pointeur du fichier associé au fichier **fp**.
    - La première lecture ou écriture ultérieure se fera à partir de la nouvelle position.

Fonctions exotiques

- Positionnement
  - Fonction : **fseek**
    - *int fseek(FILE *fp, long deplacement, int origine)*
    - Positionne le pointeur du fichier associé au fichier **fp**.
    - La première lecture ou écriture ultérieure se fera à partir de la nouvelle position.
    - Position = Valeur de **déplacement** + Valeur de base (**Origine**)
    - La valeur de cet argument **origine** doit être une des constantes (définies dans <stdio.h>) :
      - *SEEK SET : base = le début du fichier*
      - *SEEK CUR : base = la position courante*
      - *SEEK END : base = la fin du fichier*

Fonctions exotiques

- Positionnement
  - Fonction : **rewind**
    - *void rewind(FILE *fp)*
    - Positionne le pointeur au début du fichier pointé par **fp**.

Fonctions exotiques

- Positionnement
  - Fonction : **rewind**
    - *void rewind(FILE *fp)*
    - Positionne le pointeur au début du fichier pointé par **fp**.

- Autres fonctions
  - fflush
  - fgetpos
  - fsetpos
  - …