ENTREES ET SORTIES SUR FICHIERS

Plan

1. Exemple
2. Fichiers
3. Entrées et sorties standard
4. Ouverture d'un fichier
5. Ecriture dans un fichier
6. Lecture dans un fichier
7. Retour sur l'exemple
8. Redirection

Exemple

Le programme `concat`:

```
#include <stdio.h>

main (int argc, char *argv[])
{
  FILE *in, *out;
  int c;

  in = fopen (argv[1], "r"));
  out = fopen (argv[2], "a"));
  while ((c = fgetc (in)) != EOF)
    fputc (c, out);
}
```

Ainsi

```
concat source dest
```

ajoute le fichier `source` à la fin du fichier `dest`.

```
$ cat x
Bonjour, $ cat y
monde !
$ concat x hello
$ concat y hello
$ cat hello
Bonjour, monde !
$
```

Fichiers

- Un *fichier* est un fichier disque, ou le clavier ou l'écran ou tout autre périphérique.
- Toute entrée et sortie est relative à un fichier.
- L'information de gestion est contenue dans une structure de nom `FILE` et définie dans `stdio.h`.
- A tout fichier est associé une structure `FILE`.
- Toute opération sur un fichier se fait au moyen du *pointeur* sur la structure (appelé aussi *pointeur de fichier*).

Entrées et sorties standard

Au début de l'exécution d'un programme, trois structures `FILE` sont associées au terminal. Les adresses de ces structures ont pour nom `stdin`, `stdout` et `stderr`.

- `stdin` correspond à un fichier appelé *entrée standard* ouvert en lecture. Ce "fichier" est le clavier. On lit par `scanf` ou `getchar` ou par une fonction générale, en spécifiant le pointeur de fichier.
- `stdout` correspond à un fichier appelé *sortie standard* ouvert en écriture. Ce "fichier" est l'écran. On écrit par `printf` ou `putchar` ou par une fonction générale, en spécifiant le pointeur de fichier.
- `stderr` correspond à un fichier appelé *sortie erreur standard* ouvert en écriture. Ce "fichier" est aussi l'écran. On y écrit traditionnellement les messages d'erreur.

Ouverture d'un fichier

```
FILE* fopen(const char* fichier, const char* mode)
```

Par exemple

```
FILE* pf;
pf = fopen("toto", "w");
```

Les modes principaux sont

- `r` pour l'ouverture en lecture;
- `w` pour l'ouverture en écriture, avec création du fichier s'il n'existe pas et écrasement du contenu précédent s'il existe;
- `a` pour l'ouverture en ajout à la fin si le fichier existe, et création du fichier s'il n'existe pas.

En cas d'erreur, `fopen` retourne `NULL`.

Ecriture dans un fichier

On utilise `fputc` pour l'écriture d'un caractère, et `fprintf` pour l'écriture formatée.

```
fputc (c, pf);
```

écrit `c` dans le fichier de pointeur `pf`.

```
fprintf (pf, "2+2=%d\n", 4);
```

Ainsi,

```
printf(...)
```

est équivalent à

```
fprintf(stdout,...)
```

De même,

```
putc(stdout)
```

équivaut à

```
putchar()
```

Il n'y a pas de fonction prédéfinie pour l'écriture sur la sortie erreur standard.

Lecture dans un fichier

On utilise `fgetc()` pour la lecture d'un caractère et `fscanf()` pour la lecture formatée.

```
fgetc (pf);
```

retourne le caractère suivant du fichier de pointeur `pf`.

```
fscanf (pf, "%d%d\n", &i, &j);
```

lit deux entiers dans le fichier de pointeur `pf`. Ainsi,

```
scanf(...)
```

est équivalent à

```
fscanf(stdin,...)
```

De même,

```
fgetc(stdin)
```

équivaut à

```
getchar()
```

La fonction

```
ungetc(int c, FILE* pf)
```

met dans le fichier le caractère `c` de sorte qu'il soit le premier caractère lu.

Retour sur l'exemple

La prise en compte des erreurs dans le programme `concat` conduit à:

```
#include <stdio.h>

main (int argc, char *argv[])
{
  FILE *in, *out;
  int c;
  if (argc != 3) {
    fprintf (stderr,
      "%s: Usage: %s <source> <dest>\n", argv[0], argv[0]);
    exit (1);
  }
  if (!(in = fopen (argv[1], "r"))) {
    fprintf (stderr,
      "Ouverture de %s impossible en lecture\n", argv[1]);
    exit (2);
  }
  if (!(out = fopen (argv[2], "a"))) {
    fprintf (stderr,
      "Ouverture de %s impossible en ajout\n", argv[2]);
    exit (2);
  }
  while ((c = fgetc (in)) != EOF)
    fputc (c, out);
}
```

Redirection

Si, sous Unix, on lance une commande

```
com <entree >sortie 2>erreur
```

- l'entrée standard est redirigée vers le fichier `entree`,
- la sortie standard est redirigée vers le fichier `sortie`,
- la sortie erreur standard est redirigée vers le fichier `erreur`.

La fonction `freopen` permet d'associer un nouveau fichier à une structure `FILE` déjà existante.

```
FILE* freopen(const char* f, const char* m, FILE *pf)
```

Par exemple:

```
freopen("entree", "r", stdin);
freopen("sortie", "a", stdout);
freopen("erreur", "a", stderr);
```