Flux d'E/S standards : stdin, stdout, stderr

1. Les flux standard en C

Acquis : vous savez qu'un programme (par exemple C) peut communiquer avec le reste du monde au moyen de *flux* : lecture par exemple au clavier ou dans un fichier (ou ailleurs) ; et écriture par exemple dans le Terminal ou dans un fichier (ou ailleurs). Le type C qui permet de manipuler des flux est le type `FILE *`, défini dans `<stdio.h>`.

Nouveau : un programme écrit en C (mais aussi dans d'autres langages) dispose dès sa création de trois flux déjà ouverts, appelés « flux standards » :

- Le flux d'entrée standard `stdin`. Les fonctions `scanf(), getchar(), gets(),` etc. lisent dans le flux d'entrée standard. Il est par défaut connecté au clavier.
- Le flux de sortie standard `stdout`. Les fonctions `printf(), putchar(), puts(),` etc. lisent dans le flux d'entrée standard.
- Le flux d'erreur standard `stderr`.

Comme les variables `stdin, stdout et stderr` sont de type `FILE *`, elles s'utilisent comme n'importe quel autre flux, au moyen des fonctions de lecture et écriture dans les flux que vous connaissez bien : `fgets() et fscanf() ou fputs() et fprintf(),` etc. – ou encore, si on veut lire/écrire en binaire `fread() ou fwrite(),` etc.

Le programme suivant est donc tout à fait valide :

```
#include <stdio.h>
int main ( ) {
    int i;
    int nbLus = fscanf( stdin, "%d", &i) ;
    if(nbLus != 1) {
        // erreur : on n'a pas pu lire un entier
        fprintf(stderr, "Erreur : lecture entier impossible !") ;
        return 1 ; // sortie du programme
    }
    fprintf("stdout", "valeur lue %d", i) ;
    return 0 ;
}
```

Notez que par défaut, le flux d'entrée standard est dirigé depuis au clavier et les flux de sortie standard et d'erreur standard sont connecté à l'écran (au Terminal d'où est lancé le programme, en fait).

2. Quid des fonctions scanf(), printf() et consœurs ?

Vous pensez sans doute que les fonctions `scanf()` et `printf()` servent respectivement à « lire au clavier » et « écrire dans le Terminal à l'écran » depuis un programme C.

Ce n'est pas tout à fait exact. En fait :

- `scanf()` lit en fait sur le « flux d'entrée standard » du programme : `stdin ; mais`
- `printf()` écrit en fait sur le « flux de sortie standard » du programme : `stdout`

Ainsi, les fonctions `scanf()`, `printf()` et consœur sont en fait des « raccourcis » pour respectivement `fscanf(stdin, ...)` et `fprintf(stdout, ...)`, de sorte que les appels suivants sont rigoureusement équivalents :

```
scanf(.....) ;   ⇔   fscanf( stdin, .....) ;
printf(.....) ;  ⇔   fprintf( stdout, .....) ;
puts(.....) ;    ⇔   fputs( stdout, .....) ;
```

etc.

3. Usage des flux de sortie et d'erreur standard

Par convention :

- Un programme doit envoyer ses résultats dans sa sortie standard `stdout`.
- Mais il doit envoyer les messages qui ne sont pas des résultats (message de debug, messages d'erreurs...) **exclusivement** dans son erreur standard `stderr`.

Autrement dit, en C, pour écrire un *résultat* (par défaut à l'écran), on utilisera par exemple `fprintf(stdout, ...)` – ou si vous préférez `printf()`.

Par contre, pour écrire un *message d'erreur*, on utilisera **exclusivement** le flux d'erreur standard `fprintf(stderr, ...)`.

Comme on l'a indiqué, dans les deux cas, par défaut, tous les messages s'écriront à l'écran.

Pourquoi, alors, séparer les sorties valides des erreurs ?

La raison en est que, lorsqu'on exécute un programme depuis le Terminal, il est possible de *rediriger* les flux. On peut, par exemple, rediriger les résultats dans un fichier et ne garder à l'écran que les messages d'erreur ; ou ne pas afficher les messages d'erreurs (mais uniquement les résultats) à l'écran, etc.

4. Redirection des standards dans le Terminal

Cette fiche n'a pas pour objet de présenter la redirection des flux standard en Shell.

Signalons tout de même pour votre gouverne que :

1. **l'entrée standard d'un programme peut être dirigée depuis un fichier avec le symbole <**

Exemple :

```
$    sort < unFichier
```

... affiche dans le terminale une sortie triée par ordre alphabétique des lignes du fichier unFichier

2. **la sortie standard d'un programme peut être redirigée dans un fichier avec le symbole >**

Exemple :

```
$    ls *.c > listedefichiersC
```

... crée un fichier « listedefichiersC » contenant la liste des fichiers d'extension .c du répertoire courant

3. **l'erreur standard d'un programme peut être redirigée dans un fichier au moyen de 2>**

Exemple :

```
$    gcc prg.c –o out    2> /dev/null
```

... compile prg.c mais n'affiche aucun des erreurs : elles sont redirigée dans le « puit sans fond » /dev/null.

4. **la sortie standard d'un programme peut être redirigée dans l'entrée standard d'un autre programme au moyen de | . C'est ce qu'on appelle un *pipe* (ou tube, en Français) en Shell.**

Exemple :

```
$    ps –ax | grep firefox
```

... affiche les processus nommés « firefox » en cours d'exécution sur la machine.

5. Les flux de sortie et d'erreur standard dans le projet C 2A Phelma

Le script de test fourni dans le cadre du projet C 2A Phelma fait un usage intensif de la notion de flux de sortie et d'erreur standard. En particulier, il va *comparer* la sortie standard de votre programme avec la sortie attendue, pour s'assurer que tout va bien.

Il est donc essentiel que, dans le cadre du projet, votre programme réserve sa sortie standard pour les résultats et écrive bien les messages d'erreur, warning et messages de de *debugging* dans le flux d'erreur standard.

Notez que les macro `DEBUG_MSG()` etc. définies dans `notify.h` écrivent toutes bien dans l'erreur standard du programme.