Redirection

Flux

Le flux est le canal par lequel vont transiter les données entre les espaces, internes et externes, du programme.

On distingue deux catégories de flux :

- entrant: fournissant les données en entrée

- sortant: affichant les données en sortie

En C++, ces flux d'entrée et de sortie sont représentés par les 3 objets: « cin », « cout » et « cerr ».

Chaque flux ouvert en entrée ou en sortie (y compris les fichiers) se voit attribuer un numéro de descripteur.

Le tableau ci-dessous représente les flux d'entrée et de sortie dans les langages « C » et « C++ », ainsi que le numéro de descripteur associé.

C	C++	Numéro de descripteur
stdin	cin	0
stdout	cout	1
stderr	cerr	2

Principe de redirection

La redirection permet de capturer par exemple la sortie d'un fichier (ou d'une commande ou d'un programme etc.) et le renvoyer comme entrée d'un autre fichier (ou commande ou programme etc.).

Shell

Un shell (une coquille) est un interpréteur de commandes disposant d'un véritable langage de programmation. Il est considéré comme étant l'interface entre l'utilisateur et le système d'exploitation.

Les commandes à interpréter sont le plus souvent regroupées dans un fichier appelé « fichier script ».

L'interprétation des redirections dépend énormément du shell utilisé. Dans ce qui suit, nous allons décrire les commandes à utiliser pour les deux shell les plus populaires :

- bash (Bourne-again shell) est un shell Unix de la famille « open-source ». Il est utilisé comme shell par défaut sur la plus part des systèmes Linux.

- tcsh (Tenex C Shell) est une version améliorée du CShell (C Shell) dont la syntaxe ressemble à la syntaxe du langage C.

Pour connaître le « shell » utilisé par défaut dans votre « xterm », tapez la commande:

```
echo $SHELL

[/u/bozoleclown]echo $SHELL
/bin/bash
```

Redirection de la sortie vers un fichier

```
test.exe > sortie.txt
```

Cette commande permet de rediriger la sortie obtenue suite à l'exécution du programme « test.exe » vers le fichier « sortie.txt ».

Cette redirection va créer le fichier « sortie.txt » s'il n'est pas présent. Si le fichier « sortie.txt » existe déjà :

- Sous « bash » (ou « sh ») : la commande « > » va écraser le contenu de l'ancien fichier « sortie.txt ».

- Sous « tcsh » : la commande « > » va générer un message d'erreur comme quoi le fichier existe déjà. Il faudra préalablement l'effacer. On peut utiliser aussi la commande suivante pour forcer sa destruction :

```
test.exe >! sortie.txt
```

Par ailleurs, on peut aussi ajouter la sortie au fichier « sortie.txt » sans l'écraser avec la commande (valable pour sh/bash/tcsh) :

```
test.exe >> sortie.txt
```

On concatène la sortie obtenue suite à l'exécution du programme « test.exe » au contenu du fichier « sortie.txt ».

Pour l'exemple ci-dessous :

```
int main() {
    cout << "La sortie standard\n";
    cerr << "La sortie des erreurs\n";
    return 0;
}
```

On remarque que le fichier « sortie.txt » ne contient que la ligne suivante :

```
La sortie standard
```

La sortie des erreurs est quant à elle affichée dans la console.

```
test.exe > out.txt
La sortie des erreurs
```

Gestion de flux

Il existe trois flux majeurs :

1. l'entrée standard identifiée par le descripteur « 0 » ;
2. la sortie standard identifiée par le descripteur « 1 » ;
3. la sortie d'erreurs standard identifiée par le descripteur « 2 »

Redirection de la sortie des erreurs vers un fichier

Comme nous l'avons montré précédemment, par défaut les erreurs sont affichées à l'écran. Vous pouvez rediriger cet affichage vers un fichier comme suit :

Sous « sh » ou « bash », on manipule le descripteur « 2 ». On informe le système que ce descripteur est maintenant représenté par le fichier « erre.txt ». Ainsi tous les affichages envoyés vers la sortie des erreurs seront redirigés vers le fichier « err.txt ».

```
test.exe 2> err.txt
La sortie standard
```

Pour rediriger la sortie standard vers un fichier à part, il faut écrire :

```
(test.exe > sortie.txt) 2> err.txt
```

Sous « tcsh », pour rediriger l'erreur nous utilisons le symbole « & ». Sauf que ce symbole redirige à la fois la sortie standard et la sortie des erreurs. Pour séparer les deux affichages, on inclut le premier dans des parenthèses pour qu'il soit exécuté en premier. Le reste de l'affichage est envoyé par la suite vers le fichier des erreurs.

```
(test.exe > /dev/tty) >& err.txt
La sortie standard
```

Pour rediriger la sortie standard vers un fichier à part, il faut écrire :

```
(test.exe > sortie.txt) >& err.txt
```

La sortie standard et la sortie des erreurs vers un même fichier

Sous « sh » ou « bash » :

```
test.exe > latotale.txt 2>&1
```

On redirige la sortie standard vers le fichier « latotale.txt » puis la sortie des erreurs, représentée par le descripteur « 2 » vers la sortie standard qui est représentée aussi par le descripteur « 1 ». Cette sortie standard qui n'est autre que le fichier « latotale.txt ».

Sous « tcsh » :

```
test.exe >& latotale.txt
```

L'ordre des affichages en sortie

Pour l'exemple suivant :

```
int main() {
    cout << "La sortie standard\n";
    cerr << "La sortie des erreurs\n";
    return 0;
}
```

La sortie obtenue, fonction de l'environnement utilisé, est comme suit :

```
La sortie des erreurs
La sortie standard
```

On remarque que l'affichage en rapport avec « cerr » a été affiché en premier. En réalité, sur certains systèmes, le flux « cerr » n'utilise pas une zone tampon (un « buffer ») pour contenir provisoirement l'information à afficher en sortie. De ce fait, tout ce qui est envoyé vers « cerr » se retrouve immédiatement affiché en sortie. Comme ce flux représente celui des erreurs, il est donc utile d'avoir ces erreurs affichées très vite afin d'informer l'utilisateur d'un mauvais fonctionnement du programme. À noter que ce comportement peut être ajusté avec quelques commandes « C++ » afin de se comporter de la même manière que le « cout » et donc d'utiliser lui aussi une zone tampon « un buffer ».