Redirections

1. Principe

Les redirections sont l'une des plus importantes possibilités offertes par le shell. Par redirection, on entend la possibilité de rediriger l'affichage de l'écran vers un fichier, une imprimante ou tout autre périphérique, les messages d'erreur vers un autre fichier, de remplacer la saisie clavier par le contenu d'un fichier.

Tout flux de données en entrée ou en sortie de commande passe par un canal. Comme pour l'eau, il est possible de dévier le cours des données vers une autre destination ou depuis une autre source. (*ou tout faire disparaitre en redirigeant vers /dev/null*)

Linux utilise des canaux d'entrées/sorties pour lire et écrire ses données. Par défaut le canal d'entrée est le clavier, et le canal de sortie, l'écran. Un troisième canal, le canal d'erreur, est aussi redirigé vers l'écran par défaut.

Il est possible de rediriger ces canaux vers des fichiers, ou du flux texte de manière transparente pour les commandes Linux.

2. En sortie

On se sert du caractère **>** pour rediriger la sortie standard (celle qui va normalement sur l'écran). On indique ensuite le nom du fichier où seront placés les résultats de sortie.

```
$ ls -l > resultat.txt
$ cat resultat.txt
total 1
-rw-r--r--   1 Administ ssh_user       0 Jul  4 12:04 TOTO
-rw-r--r--   1 Administ ssh_user       0 Jul 25 15:13 resultat.txt
-rw-r--r--   1 Administ ssh_user     171 Jul 25 15:13 test.txt
```

Si le fichier n'existe pas, il sera créé. S'il existe, son contenu sera écrasé, même si la commande tapée est incorrecte. **Le shell commence d'abord par créer le fichier puis exécute ensuite la commande**.

C'est un aspect important des redirections : **les redirections sont interprétées de la droite vers la gauche, et les redirections sont mises en place AVANT l'exécution des commandes** : il faut bien créer le fichier avant de pouvoir y écrire. D'où le fait que même si la commande est fausse, le fichier est créé ou écrasé...

Pour ajouter des données à la suite du fichier, donc sans l'écraser, on utilise la double redirection **>>**. Le résultat de la commande est ajouté à la fin du fichier.

```
$ ls -l > resultat.txt
$ date >> resultat.txt
$ cat resultat.txt
total 1
-rw-r--r--    1 Administ ssh_user       0 Jul  4 12:04 TOTO
-rw-r--r--    1 Administ ssh_user       0 Jul 25 15:13 resultat.txt
-rw-r--r--    1 Administ ssh_user     171 Jul 25 15:13 test.txt
Thu Jul 25 15:20:12  2002
```

3. En entrée

Les commandes qui attendent des données ou des paramètres depuis le clavier peuvent aussi en recevoir depuis un fichier, à l'aide du caractère <. Un exemple avec la commande **wc** (word count) qui permet de compter le nombre de lignes, de mots et de caractères d'un fichier.

```
$ wc  < resultat.txt
    4     29    203
```

4. Documents en ligne

La redirection << est particulière. Elle permet l'édition des documents en ligne. Vous trouverez parfois le terme Herescript ou Here Document. Cela permet la saisie d'un texte jusqu'à un point donné et l'envoi de son résultat à une commande ou un filtre. Les redirections classiques sont aussi autorisées. Après le << vous indiquez une chaîne définissant la fin de saisie, par exemple ici 'end'.

```
$ tr "[a-z]" "[A-Z]" << end
> bonjour les amis
> ceci est un exemple
> de herescript
> end
BONJOUR LES AMIS
CECI EST UN EXEMPLE
DE HERESCRIPT
```

Une variante existe, appelée le Here String, utilisant la redirection <<<. Contrairement à <<, elle n'a pas besoin de délimiteur de fin.

```
$ cat <<< toto
toto
```

Le plus intéressant est l'utilisation couplée avec une substitution de commandes (nous l 'aborderons avec la programmation shell) :

```
$ tr "[a-z]" "[A-Z]" <<< $(cat fictest)
BONJOUR LES AMIS CECI EST UN EXEMPLE DE HERE STRING
```

5. Les canaux standards

On peut considérer un canal comme un fichier, qui possède son propre descripteur par défaut, et dans lequel on peut ou lire ou écrire.

- Le canal d'entrée standard se nomme **stdin** et porte le descripteur **0**.
- Le canal de sortie standard se nomme **stdout** et porte le descripteur **1**.
- Le canal d'erreur standard se nomme **stderr** et porte le descripteur **2**. On peut rediriger le canal d'erreur vers un autre fichier.

```
$ rmdir dossier2
rmdir: `dossier2': No such file or directory
$ rmdir dossier2 2>error.log
$ cat error.log
rmdir: `dossier2': No such file or directory
```

Vous pouvez rediriger les deux canaux de sortie dans un seul et même fichier, en les liant. On utilise pour cela le caractère **>&**. Il est aussi important de savoir dans quel sens le shell interprète les redirections. Les redirections étant en principe en fin de commande, le shell recherche d'abord les caractères **<**, **>**, **>>** en fin de ligne. Ainsi si vous voulez grouper les deux canaux de sortie et d'erreur dans un même fichier, il faut procéder comme suit.

```
$ ls -l > resultat.txt 2>&1
```

La sortie 2 est redirigée vers la sortie 1, donc les messages d'erreurs passeront par la sortie standard. Puis le résultat de la sortie standard de la commande **ls** est redirigé vers le fichier resultat.txt. Ce fichier contiendra donc à la fois la sortie standard et la sortie d'erreur. Une autre syntaxe, au résultat identique, est possible :

```
$ ls -l &>resultat.txt
```

Vous pouvez utiliser les deux types de redirection à la fois :

```
$ wc < resultat.txt > compte.txt
$ cat compte.txt
    4     29    203
```

Pour écrire dans le canal d'erreur avec un echo, faites comme ceci : redirigez la sortie standard vers le canal d'erreur.

```
$ echo "erreur" >&2
```

6. Ouverture de canaux

Les canaux standards sont au nombre de trois et numérotés de 0 à 2. Ainsi 0< équivaut à < et 1> à >. La commande **exec** permet d'ouvrir sept autres

canaux numérotés de 3 à 9. On a donc en tout dix canaux.

Vous pouvez, et même devez, envisager dans le cadre de traitements de sortir certains résultats par le canal 3, d'autres par le 4, et ainsi de suite. Les canaux ouverts le sont en entrée et en sortie.

```
$ exec 3>dump.log
$ ls -l >&3
$ cat dump.log
total 3952
-rw-r--r--  1 driss users  167212 oct  9 09:27 brise_1280.jpg
drwxr-xr-x  2 driss users    4096 mar  4 08:51 bin
drwxr-xr-x  8 driss users    4096 mar  4 08:45 cxoffice
drwx------  2 driss users    4096 mar 10 12:29 Desktop
drwx------ 13 driss users    4096 mar  6 11:49 Documents
-rw-r--r--  1 driss users       0 mar 11 11:34 dump.log
-rw-r--r--  1 driss users 3785296 déc 12 15:15 e3555_mcdMaisonDuMonde.pdf
drwxr-xr-x  3 driss users    4096 mar 10 11:16 Games
drwxr-xr-x  5 driss users    4096 mar 10 11:16 karchiver-3.4.2.b4
-rw-r--r--  1 driss users     358 mar 11 08:51 liste
-rw-r--r--  1 driss users     608 mar 11 09:14 tmpgrp
-rw-r--r--  1 driss users    1555 mar 11 09:15 tmppwd
```

Tous ce qui sera écrit dans le canal 3 , qui en fait sera écrit dans le fichier dump.log. On peut ensuite fermer le canal en le réunissant avec un pseudo-canal (canal de fermeture -).

```
$ exec 3>&-
```

7. Filtre : définition

Un **filtre** (ou une commande filtre) est un programme sachant écrire et lire des données par les canaux standards d'entrée et de sortie. Il en modifie ou traite éventuellement le contenu. wc est un filtre. En voici quelques-uns : **more** (affiche les données page par page), **sort** (tri des données), **grep** (critères de recherche).

8. Pipelines / tubes

Les redirections d'entrée/sortie telles que vous venez de les voir permettent de rediriger les résultats vers un fichier. Ce fichier peut ensuite être réinjecté dans un filtre pour en extraire d'autres résultats. Cela oblige à taper deux lignes : une pour la redirection vers un fichier, l'autre pour rediriger ce fichier vers le filtre. Les **tubes** ou **pipes** permettent de rediriger directement le canal de sortie d'une commande vers le canal d'entrée d'une autre. Le caractère permettant cela est **|**, accessible depuis la combinaison [AltGr] **6** des claviers PC standards français, ou [Alt][Shift] l (lettre L) des claviers Mac.

```
$ ls -l > resultat.txt
$ wc < resultat.txt
```

devient

```
$ ls -l | wc
```

Il est possible de placer plusieurs | sur une même ligne.

```
$ ls -l | wc | wc
      1       3      24
```

Quand vous réaliserez vos script, la première commande n'est pas forcément un filtre. L'essentiel est qu'un résultat soit délivré. Idem pour la dernière commande qui peut par exemple être une commande d'édition ou d'impression. Enfin, la dernière commande peut elle-même faire l'objet d'une redirection en sortie.

```
$ ls -l | wc > resultat.txt
```

TRAVAUX PRATIQUES

1. find - Rechercher des fichiers dans une hiérarchie de répertoires. La commande **find /** retourne beaucoup d'erreurs si elle est utilisée par un simple utilisateur à cause d'un problème de droits. Évitez les messages d'erreurs en les redirigeant vers un « trou noir » :

   ```
   $ find / 2>/dev/null
   ```

2. Dans le cas précédent et malgré les erreurs, vous avez encore accès à beaucoup d'emplacements et la liste qui s'affiche est très longue et donc inexploitable. Placez cette liste dans un fichier appelé résultat.

   ```
   $ find / 1>resultat 2>/dev/null
   ```

3. Maintenant, plus rien ne s'affiche. En fin de compte, pour savoir pourquoi vous ne pouvez pas accéder à certains répertoires vous voulez aussi obtenir les messages d'erreurs dans le fichier résultat, avec la liste des fichiers. Faites une redirection du canal d'erreur standard dans le canal de sortie standard :

   ```
   $ find / >resultat 2>&1
   ```

4. Plus rien ne s'affiche. Vous voulez les deux : un fichier et l'affichage des résultats sur écran. La commande **tee** s'utilise avec un tube et permet de récupérer un flux sortant, de le placer dans un fichier et de ressortir ce flux comme si de rien n'était :

```
$ find / 2>&1 | tee resultat.
```

Votre fichier résultat contient la liste de tous les fichiers accessibles, les erreurs et le tout s'affiche aussi sur l'écran.

EXERCICES

Les redirections

1 -Quel est le résultat de la commande **>fic** sans rien devant ?

2- Que va afficher **ls -R / > liste** et que contiendra liste si vous exécutez cette commande avec un utilisateur ordinaire ?

3- En tant qu'utilisateur, pour lister l'intégralité des fichiers du système et placer tous les résultats quels qu'ils soient dans liste, quelle doit être la bonne redirection ?

- A - 2>dev/null >liste
- B - >liste 2>liste
- C - 1>liste 2>&1
- D - 2>1 >liste

4- Comment via echo faire passer un message par le canal d'erreur ?

5- Comment faire afficher le contenu d'un fichier avec cat et une redirection en entrée ?

6- Comment supprimer tous les messages d'erreur ?

- A - 2>/dev/null
- B - 2>/dev/zero
- C - >/dev/null 2>&1
- D - 2>fic ; rm fic