Redirections - Pipes - Commandes récursives

Objectifs

— Interpréteur de commandes : shell,
— Redirections,
— Tubes.

1 Redirections

Les calculs en cours (ou processus) échangent des informations avec leur environnement (périphériques, fichiers, ...) sous la forme de suites d'octets (ou flots d'entrées/sorties).

Initialement, le shell dispose de trois flots d'entrée/sortie :
— l'entrée standard (flot numéro 0), stdin, associée au clavier,
— la sortie standard (flot numéro 1), stdout, associée à l'écran,
— la sortie erreur standard (flot numéro 2), stderr, associée aussi à l'écran.
Ces associations des flots standard sont conservées, par défaut pour les commandes lancées à partir du shell. Néanmoins, il est possible de redéfinir ces associations, pour une commande donnée : on parle alors de redirection d'un flot. Ainsi :

— la sortie standard peut être définie à l'aide du caractère `>` :
`ls -l > liste` : permet d'écrire le résultat de `ls` dans le fichier *liste*,

— il est possible de conserver le contenu d'un fichier vers lequel on redirige la sortie standard :
`ls -l >> liste` : ajoute le résultat de `ls` à la fin du fichier *liste*,

— en Bourne Shell (`sh` ou `bash`), la sortie erreur standard peut être redirigée par `2>`
`rm * 2> log` : écrit les messages d'erreurs éventuellement engendrés par l'exécution de `rm` dans le fichier *log*,

— l'entrée standard peut être redirigée par `<`
`sort < tolkien.txt` classe les lignes du fichier tolkien.txt par ordre alphabétique.

2 Tubes

La sortie standard d'une commande peut être reliée à (redirigée vers) l'entrée standard d'une autre commande par un *tube* (ou *pipe* en anglais). La seconde commande peut alors utiliser comme données d'entrée de son traitement les résultats fournis par la première commande. Ce schéma de coordination est appelé un schéma *producteur/consommateur* et sera revu longuement lorsque vous suivrez les cours

de Systèmes Centralisés.

L'utilisation d'un tube permet de transmettre un flot de données entre deux commandes sans avoir à utiliser un fichier intermédiaire : les données transmises sont conservées dans un tampon, en mémoire vive sans que l'utilisateur en ait connaissance. De plus, la synchronisation des deux commandes est implicite.

La mise en place d'un tube est définie à l'aide du caractère `|` obtenu en appuyant simultanément sur les touches «Alt Gr» et «-» («6»).

`ls -l | more` : relie la sortie de `ls -l` à l'entrée de `more`. La liste des fichiers du répertoire courant sera traitée par `more`, c'est-à-dire affichée page à page (bien utile quand le nombre de fichiers contenus dans un répertoire est important).

Notez bien que la commande `more` s'applique sur le flot de données composé par les noms des fichiers et non sur les fichiers eux même. Notez la différence entre les deux lignes de commandes :

- `ls *.txt | more`
- `ls *.txt | xargs more`

Essayez aussi : `set | more` pour lister les variables d'environnement.

3 Commandes récursives

Les principales commandes de manipulation de l'arborescence des fichiers possèdent une option récursive `-R` qui s'appliquent à un répertoire et à tous ses sous-répertoires :

- `ls -R` : liste récursivement dans un parcours «en largeur» un répertoire et ses sous-répertoires,

- `cp -R rep1 rep2` : fait une copie récursive du répertoire rep1
  - dans un nouveau répertoire de chemin d'accès rep2, si rep2 n'existe pas,
  - dans un sous-répertoire de nom rep1, dans le répertoire rep2 s'il existe déjà !

- `rm -R rep` : détruit tout le répertoire rep et son contenu récursivement ! Utiliser avec prudence ou avec l'option `-i`.

Travail à effectuer :
Expérimenter les options récursives sur l'archive.

4 Deux commandes shell bien utiles

`grep [options] exp [f1 ...]`

Affiche toutes les lignes des fichiers f1... contenant l'expression régulière `exp`. Si la commande porte sur plusieurs fichiers, chaque ligne affichée est précédée du nom de fichier la contenant.

Quelques options intéressantes :

- `-i` rend la recherche insensible à la "casse" des lettres (minuscules ou MAJUSCULES)
- `-n` indique la ligne où l'expression a été trouvée

- `-l` ne donne que le nom des fichiers où la recherche a été fructueuse (utile par exemple pour éditer les fichiers qui contiennent l'expression recherchée)

```
grep alias $HOME/.bashrc
grep begin *.tex
grep -in The tolkien.txt
```

Remarques :
— il convient, bien entendu, de se placer dans un répertoire qui contient les « bons » fichiers,
— la variable globale `HOME` (que nous reverrons au dernier TP) contient le chemin de votre répertoire de connexion.
— `exp` peut contenir des métacaractères permettant de décrire des motifs de recherche. Ces métacaractères sont spécifiques à la commande `grep` et sont différents de ceux du SHELL.

`find repertoires expression...`

Parcourt récursivement les répertoires spécifiés et évalue, pour chaque fichier l'expression de gauche à droite. L'expression permet soit de définir des critères de sélection, soit d'effectuer des opérations (affichages, exécution de commandes) sur les fichiers vérifiant les critères de sélection précédents.

```
find . -name '*.txt' -print
```

affiche (option `-print`) tous les fichiers de l'arborescence du répertoire courant (`.`) qui ont comme suffixe `*.txt` (option `-name '*.txt'`).

```
find $HOME -name '*.txt' -print -exec grep -i '^the'  {} \;
```

trouve tous les fichiers de l'arborescence de votre répertoire de connexion ($HOME) dont le suffixe est `*.txt` et pour ces fichiers, affiche leur nom puis, affiche les lignes contenant l'expression `the` en début de ligne (option `-exec` avec la commande `grep -i '^the'`).

Remarques :
— "{}" désigne le nom des fichiers trouvés précédemment par le `find`,
— le " ;" (précédé de son \) marque la fin du bloc `-exec`,
— dans le dernier exemple, les affichages du `-print` et celui du `grep` sont effectués à la suite l'un de l'autre.

Une dernière utilisation de la commande `find` avec en prime un tube et la commande `xargs` :

```
find $HOME -name '*.txt' -print | xargs grep -i '^the'
```

Cette commande donne le même résultat que la précédente et on se passe des accolades et de l'antislash-point virgule. Elle fonctionne comme suit :

— le flux résultat de find est en entrée de la commande `xargs`,
— celle-ci exécute autant de fois la commande en argument (ici `grep`) qu'il y a d'éléments en entrée,
— chacun de ces éléments devenant le dernier argument de la commande `grep`.

5 Un petit test rapide

Dans votre dossier de travail, créez un dossier TEST qui contiendra 3 sous-dossiers TEST1, TEST2 et TEST3. Chaque sous-dossier TESTi contiendra 3 sous-dossiers TESTB1, TESTB2, et TESTB3.

Dans chacun de ces 9 dossiers, il y a 3 fichiers avec l'extension .txt. Chacun de ces fichiers contient 5 lignes qui commencent par le mot Une et 5 lignes qui commencent par le mot une. 5 de ces lignes contiennent le mot test.

Attention ! Il ne s'agit pas de créer tous les dossiers un par un, tous les fichiers un par un, ou toutes les lignes une par une. Pensez aux commandes de copie de lignes, de substitution de mots, de copie de fichiers, de copie de dossiers.

1. écrivez la commande qui affiche les lignes commençant par Une dans tous les fichiers .txt du dossier TEST et de ses sous-dossiers
2. écrivez la commande qui affiche les lignes contenant le mot test dans tous les fichiers .txt du dossier TEST et de ses sous-dossiers
3. regardez ce que fait la commande wc
4. écrivez la commande qui compte le nombre de lignes commençant par Une dans tous les fichiers .txt du dossier TEST et de ses sous-dossiers
5. écrivez la commande qui compte le nombre de lignes commençant par Une ou par une dans tous les fichiers .txt du dossier TEST et de ses sous-dossiers
6. écrivez la commande qui compte le nombre de lignes contenant le mot test dans tous les fichiers .txt du dossier TEST et de ses sous-dossiers
7. écrivez la commande qui compte le nombre de dossiers contenus dans le dossier TEST et ses sous-dossiers
8. redirigez les sorties de ces commandes vers des fichiers

Remarques : Ne jamais appeler une commande ou un programme "test" car SHELL contient déjà une commande qui porte ce nom et qui sera exécutée prioritairement avant votre commande ou votre programme.

6 CQFAR (Ce Qu'il Faut Avoir Retenu)

À la fin de cette séance de TP vous devez :

1. savoir copier, déplacer, renommer, supprimer des fichiers et des répertoires,
2. savoir utiliser les caractères jokers,
3. savoir rediriger la sortie standard,
4. savoir coupler des commandes au moyen de tubes,
5. savoir utiliser `grep` et `find` dans des cas simples.