ISS - Initiation aux Systèmes d'exploitation et au Shell LU2IN020

TP 03 – Redirection et variables d'environnement

Julien Sopena

octobre 2022

Le but de cette troisième semaine est d'approfondir l'étude des arborescences de processus, d'introduire la notion de variables d'environnement. Nous étudierons aussi les différents types de redirections, ainsi que la substitution de commandes.

Exercice 1 : Un classique

Cet exercice repose sur la redirection des entrées/sorties des processus. Dans un système Unix, les processus peuvent lire et écrire dans les fichiers qu'ils ouvrent, mais ils peuvent aussi utiliser trois flux standard : `stdin`, `stdout` et `stderr`. Par défaut, ces flux correspondent au clavier et à l'affichage du terminal.

S'il lance simplement une commande, le père transmet ses 3 flux à son fils. Mais il peut aussi les modifier en ajoutant à la fin de la ligne de commande un ou plusieurs des symboles suivants :

- `cmd > fichier` la commande écrira dans le fichier en l'écrasant s'il existe ;
- `cmd >> fichier` la commande écrira à la fin du fichier en le créant si nécessaire ;
- `cmd 2> fichier` la commande écrira ses erreurs dans le fichier en l'écrasant s'il existe ;
- `cmd 2>> fichier` la commande écrira ses erreurs à la fin du fichier en le créant si nécessaire ;
- `cmd < fichier` la commande ne lira pas sur le clavier, mais dans le fichier.

Question 1

Pour commencer, vous allez récupérer les données nécessaires à l'ensemble des exercices. Placez-vous dans le répertoire correspondant à ce TP pour y extraire l'archive après l'avoir chargée à l'URL suivante : http ://julien.sopena.fr/LU2IN020-TP_03.tgz

Question 2

Ouvrez le fichier `cesar.c` contenu dans le répertoire `exo1` et étudiez-le. Que fait-il ?

Question 3

Testez-le après l'avoir compilé. Pour quitter le programme, il faut simuler une fin de fichier en combinant les touches `Ctrl+d`.

Question 4

Le répertoire `fenetre_sur_coquillage` contient un texte chiffré par décalage, puis fragmenté en plusieurs fichiers. En essayant tous les décalages possibles sur le premier fragment `part1`, trouvez la clé de déchiffrement.

Question 5

Maintenant que vous avez récupéré la clé, utilisez-la pour déchiffrer l'ensemble des fichiers et concaténez les résultats dans un même fichier `news.txt` qui contiendra tout le texte en clair.

Exercice 2 : SplitStrip

Pour passer les limites de certains protocoles réseau, non destinés à l'origine à l'échange de grosses données, il est courant de découper ses fichiers en petits morceaux (appelés *chunks*) avant de les mettre en ligne.

Dans cet exercice, on s'intéresse à un document disponible au téléchargement sous forme de 100 *chunks* aux URL suivantes, où `xx` est un identifiant compris entre 0 et 99 : http ://julien.sopena.fr/chunks/data.xx

Question 1

Pour commencer écrivez un script `téléchargement.sh` qui permet de télécharger l'ensemble de ces chunks. Votre script devra :

— s'assurer que le répertoire où vous allez enregistrer les chunks existe et le créer si tel n'était pas le cas ;
— ne pas recharger un chunk si celui-ci existe déjà ;
— s'arrêter si le chargement d'un fichier a échoué.

Outre ce que vous avez appris pendant ces premières semaines, vous pouvez utiliser les éléments suivants :

— `-f` de la commande test qui permet de tester l'existence d'un fichier ;
— le not `!` utilisable dans un test ou devant une commande ;
— le mot clé `break` qui permet de sortir d'une boucle.

Question 2

Maintenant que vous avez récupéré tous les chunks, vous pouvez reconstituer le fichier en concaténant tous les fichiers dans l'ordre croissant des identifiants.

Question 3

Vous venez maintenant d'obtenir le fichier complet. Si vous affichez son contenu, vous verrez qu'il s'agit d'un binaire. Pour connaître son type, vous pouvez utiliser la commande `file` qui utilise une base de signatures pour identifier les fichiers. Renommez le fichier en conséquence et ouvrez-le.

Question 4

Attention, l'abus de *Bash* peut être très dangereux pour vos fichiers : exécuter la commande `rm -rf /` effacera tous les fichiers qui vous appartiennent ou sur lesquels vous avez des droits en écriture.

Ce script repose sur l'utilisation des séparateurs de commandes `&&` et `||` qui retournent 0 si les deux (resp. une des deux) commandes se sont correctement terminées (ont retourné 0). Comme dans de très nombreux langages, l'évaluation de ces opérateurs est dite *progressive*, *i.e.*, que pour un `&&` la commande de droite ne sera exécutée que si la commande de gauche a bien fonctionné, dans le cas contraire elle peut déjà retourner une valeur différente de 0.

Proposez une version soft de ce script qui :

— affiche `"You lost everything"` au lieu de faire `rm -rf /`
— utilise des structures de contrôles classiques.
— utilise la notation `$(( ))` puisqu'elle remplace `$[ ]` aujourd'hui obsolète ;

Exercice 3 :

Question 1

La commande `wc` permet, entre autres, de compter le nombre de caractères, de mots et de lignes de fichiers dont les noms sont passés en paramètre. En absence de paramètres, cette commande mesure ce qu'elle lit sur son entrée standard. Dans ce dernier cas, seules la ou les mesures sont affichées puisqu'il n'y a pas de nom de fichier.

Proposez une commande qui permet d'afficher le nombre de caractères d'un fichier et uniquement ce nombre (pas d'autres métriques, pas de nom de fichier). Testez votre solution pour afficher 42 le nombre de caractères du fichier `test` fourni dans l'archive.

Question 2

Utilisez cette commande pour implémenter un script `biggest.sh` qui affiche le nom du plus gros fichier d'un répertoire (supposé non vide) dont le nom est passé en paramètre. Votre script devra préalablement vérifier la présence du paramètre et l'existence du répertoire.

Question 3

Créer un répertoire `sélection` au même niveau que le répertoire `dico` fourni dans l'archive.

Puis utilisez votre script `biggest.sh` pour implémenter un script `select.sh` qui déplace (sans changer leur nom) les 4 plus gros fichiers d'un répertoire (premier paramètre) dans un autre répertoire (deuxième paramètre). À la fin de votre script, le répertoire destination ne devra comporter que ces 4 fichiers, les autres n'auront pas été déplacés.

Vous pouvez tester votre script sur une copie du répertoire `dico`, faite avec la commande suivante :

```
moi@pc /home/moi $ cp -r dico dico_copie
```

Question 4

Affichez, après avoir utilisé votre script sur la version originale du répertoire dico, le contenu du répertoire contenant les 4 plus gros fichiers avec la commande `ls`.