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

TD 04 – Combinaison de commandes avec les tubes

Julien Sopena

octobre 2022

Le but de cette quatrième 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 et reviendrons sur la substitution de commandes introduite la semaine précédente.

Exercice 1 : Retour aux sources

Question 1

Pour commencer, on considère les trois scripts `A.sh`, `B.sh` et `C.sh` donnés ci-dessous. Quel affichage est produit par l'exécution du script `A.sh` ? Pourquoi ?

`A.sh`

```
#! /bin/bash

var="val_de_a"
echo "A: $var"
./B.sh
echo "A: $var"
./C.sh
echo "A: $var"
```

`B.sh`

```
#! /bin/bash

export var="val_de_b"
echo "B: $var"
./C.sh
echo "B: $var"
```

`C.sh`

```
#! /bin/bash

echo "C: $var"
```

Question 2

Expliquez, grâce à l'exemple précédent, le scénario suivant :

> *"Un étudiant ouvre une console, installe un logiciel et ajoute à son `$PATH` le répertoire qui contient l'exécutable. Il fait un test ... tout fonctionne. Un peu plus tard, il rouvre une console pour y lancer le programme. Ça ne marche plus ..."*

Question 3

Pourrait-on régler ce problème en écrivant un script que l'on exécuterait à chaque démarrage d'une console ?

Question 4

La solution vient avec le mot clé `source` et son alias `.` qui permet d'exécuter un script dans un autre sans lancer de nouveau processus. On peut le voir comme un équivalent *Bash* du `##include` du C.

Pour bien comprendre, revenons à l'exemple précédant en modifiant `A.sh` pour qu'il `source` le script `B.sh` Quel est le nouvel affichage ? Justifiez votre réponse avec un chronogramme.

A.sh

```
#! /bin/bash

var="val_de_a"
echo "A: $var"
source B.sh
echo "A: $var"
./C.sh
echo "A: $var"
```

B.sh

```
#! /bin/bash

export var="val_de_b"
echo "B: $var"
./C.sh
echo "B: $var"
```

C.sh

```
#! /bin/bash

echo "C: $var"
```

Question 5

Chaque *Bash* interactif, *i.e.*, sans script à exécuter en paramètre, va au démarrage lancer la commande `source ~/.bashrc`. Ce fichier, souvent vu à tort comme un fichier de configuration, est en fait un script comme les autres. Vous pouvez donc y placer toutes les commandes (affichage, boucle, test, ...) étudiées dans cette UE.

Quelles variables d'environnement pourrait-il être utile de redéfinir dans un `.bashrc` ?

Exercice 2 : Les tubes

Question 1

Pour commencer, implémenter un script `majuscule.sh` qui affiche les lignes lues sur son entrée standard avec uniquement des lettres en majuscule. Vous pouvez utiliser la fonctionnalité du *Bash* qui assure un passage en majuscule de son contenu d'une variable lorsqu'elle est utilisée avec la syntaxe suivante : `${var^^}`. Notons que `${var,,}` passe le contenu en minuscule.

Question 2

Implémenter un deuxième script `numerotation.sh` qui affiche les lignes lues sur son entrée standard en les préfixant par un numéro de ligne.

Question 3

Proposez une première version d'un script qui numérote et affiche le contenu en majuscule d'un fichier passé en paramètre. Ce script utilisera un fichier temporaire.

Question 4

Quelle est la limite de cette approche de composition utilisant un fichier temporaire ?

Question 5

Heureusement, le *Bash* fournit un opérateur de composition `|` qui permet de relier directement la sortie standard d'une application à l'entrée standard d'une autre. Cet opérateur repose sur une fonctionnalité offerte par le noyau Linux appelée *pipe*.

L'opérateur `|` agit donc comme un séparateur de commandes et on peut relier autant de commandes que nécessaire :

```
cmd1 | cmd2 | cmd3 | cmd4 ;
```

Proposez une nouvelle implémentation de votre script qui optimise son fonctionnement en utilisant un `pipe`.

Exercice 3 : La boîte à outils

Dans cet exercice, nous allons composer des commandes classiques de traitement de flux pour réaliser des traitements plus complexes. L'ensemble des commandes utilisées ici ont pour point commun de pouvoir traiter des fichiers passés en paramètre ou de traiter les chaînes lues sur leur entrée standard si aucun nom de fichier na été donné.

Vous allez ainsi utiliser les commandes suivantes :

`wc` : affiche, dans cet ordre, le nombre de lignes (option `-l`), de mots (option `-w`), de caractères (option `-m`) et/ou d'octets (option `-c`) ;

`head -n x` : affiche les `x` premières lignes, ou tout sauf les `x` dernières lignes si `x` est précédé par un `-` ;

`tail -n x` : affiche les `x` dernières lignes, ou tout à partir de la x^e lignes si `x` est précédé par un `+` ;

`sort` : trie les lignes par ordre alphabétique ou numérique avec l'option `-n`. Le tri, croissant par défaut, devient décroissant avec l'option `-r` ;

`uniq` : élimine les lignes consécutives dupliquées, ce qui implique de trier l'entrée avant de l'utiliser ;

`grep motif` : afficher les lignes contenant un motif donné, ou celle ne contenant pas ce motif avec l'option `-v` ;

`cut` : Extrait des parties des lignes de deux façons différentes :

- elle affiche les caractères sélectionnés en fonction de leurs positions avec l'option `-c`. Par exemple les 2^e, 4^e, 5^e et 6^e caractères avec la commande `cut -c 2,4-6`
- elle découpe la ligne suivant le caractère défini avec `-d` (tabulation par défaut) et affiche les éléments sélectionnés avec l'option `-f`. Par exemple les 2^e, 4^e, 5^e et 6^e mots avec la commande `cut -d ' ' -f 2,4-6`

Dans la suite, on considère le fichier `notes.lst` qui utilise des tabulations comme séparateur.

```
moi@pc /home/moi $ cat notes.lst
Prenom  Nom       F/G : UE1 UE2 UE3
Yuka    Nakagawa   F  :  16  12  17
Chisato Matsui     F  :  19   7   7
Haruka  Tanizawa   F  :  12   8   8
Satomi  Noda       F  :  17   5   3
Yukie   Utsumi     F  :   2  12   3
Yûko    Sakaki     F  :   9  14  15
Hiroki  Sugimura   G  :  11  12  13
Kayoko  Kotohiki   F  :   8  12  11
Yûko    Kotohiki   F  :  10   6   8
Mitsuko Sôma       F  :  11  18  18
Yutaka  Seto       G  :  12   7  12
Keita   Iijima     G  :   2   8   3
Shinji  Mimura     G  :  19  10  15
Kazuo   Kiriyama   G  :   6   7   2
Shôgo   Kawada     G  :   5   9   8
```

Question 1

Pour commencer, proposez une ou plusieurs commandes permettant d'afficher le nom des différentes colonnes.

Question 2

Proposez une ou plusieurs commandes permettant d'afficher toutes les notes de Satomi.

Question 3

Proposez une ou plusieurs commandes permettant d'afficher toutes les notes du dernier étudiant

Question 4

Proposez une ou plusieurs commandes permettant d'afficher toutes les notes du premier étudiant

Question 5

Proposez une ou plusieurs commandes permettant d'afficher le nombre de prénoms différents dans la liste.

Question 6

Proposez une ou plusieurs commandes permettant d'afficher un message indiquant si la parité est respectée dans cette promotion.

Question 7

Proposez une ou plusieurs commandes permettant d'afficher la moyenne générale de Satomi sachant qu'il y a trois notes de même coefficient.

Question 8

Proposez une ou plusieurs commandes permettant d'afficher la moyenne de la promotion à la première UE sans connaître a priori le nombre de notes.

Question 9

On veut maintenant afficher la moyenne pour chacune des UE. Pour éviter de dupliquer le code (le copier/coller en programmation est presque toujours la marque d'une erreur de conception), commencer par écrire un script qui affiche la moyenne des notes lues sur son entrée standard à raison d'une note par ligne. Utilisez ensuite ce premier script pour afficher les moyennes demandées.