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

TD 03 – Les points d'entrée pour les données d'un script

Julien Sopena

octobre 2022

Le but de cette troisième semaine est d'approfondir l'étude des arborescences de processus commencée la semaine précédente. Il aussi l'occasion de faire le tour des différentes sources d'entrée pour les données d'un script : paramètres, variables d'environnement, redirections du flux d'entrée standard et substitutions de commandes.

Exercice 1 : Du père au fils

Dans cet exercice, nous allons considérer l'exécution du code des scripts suivants :

pere.sh

```
#! /bin/bash

chaine="+"
echo "$$ : $chaine"
./fils.sh
echo "$$ : $chaine"
```

fils.sh

```
#! /bin/bash

nb=4
while [ $nb -gt 0 ] ; do
  chaine="$chaine$chaine"
  echo "$$ : $chaine"
  nb=$((nb-1))
done
```

Question 1

Pour commencer, tracez le chronogramme correspondant à l'exécution du script `pere.sh`.

Question 2

Que se passerait-il si l'on remplaçait la commande `echo` par `/bin/echo` ? Illustrez votre réponse par un nouveau chronogramme.

Question 3

Comment peut-on savoir qu'une commande est une commande built-in ?

Question 4

Quel affichage est produit par cette exécution ? D'où vient le problème ?

Question 5

En utilisant les mécanismes étudiés lors des deux premières semaines, proposez une correction de ces deux scripts qui permette au fils de faire un affichage reprenant le contenu de la variable `$chaine` du père.

Question 6

Afin de pouvoir transmettre implicitement une variable à un processus fils, les systèmes Unix introduisent le concept de *variables d'environnement*. Ainsi au moment de sa création, un fils hérite de son père toutes les valeurs de ces variables. Par convention, le nom de ces variables est en majuscule.

Dans un script, on peut à tout moment :
— Ajouter une variable déjà existante à son environnement : `export MA_VARIABLE`
— Ajouter une nouvelle variable à son environnement : `export MA_VARIABLE="valeur"`
— Retirer une variable de son environnement : `unset MA_VARIABLE`
Utilisez les variables d'environnement pour proposer une nouvelle implémentation des scripts `pere.sh` et `fils.sh`. Est-ce que cela change l'affichage du père ?

Question 7

Quel exemple de variable d'environnement avons-nous rencontré lors des deux premières semaines ?

Question 8

Pour finir, notons que l'on peut créer une variable d'environnement pour son fils sans pour autant l'exporter dans son père. Il suffit de la déclarer sur la même ligne juste avant la commande. Quel est l'intérêt de cette méthode illustrée par l'exemple suivant ?

```
moi@pc /home/moi $ HTTP_PROXY=http://monproxy.com:8080 firefox
```

Exercice 2 : Poésie et récursivité

Dans cet exercice, on veut lancer une succession de scripts, suivant le schéma suivant. Le script `jonathan.sh` va lancer le script `pelican.sh`, qui à son tour va lancer le script `pelican.sh`, et ainsi de suite ... cela pourrait durer pendant très longtemps, alors on se limitera à n d'instances de `pelican.sh`, où n sera un paramètre du script `jonathan.sh`.

Question 1

Pour commencer, tracez le chronogramme correspondant à l'exécution du script `jonathan.sh`. Pour simplifier, vous n'y représenterez que les scripts que vous avez vous-même écrits.

Question 2

On veut que le script `jonathan.sh` affiche un message lorsque tous les `pelican.sh` se sont exécutés. Est-ce possible ? Si oui comment ? Si non pourquoi ?

Question 3

Proposez deux implémentations différentes de ces scripts. La première reposera sur le passage de paramètres. Vous veillerez alors à vérifier leur présence dans le script `jonathan.sh`. La seconde, quant à elle, utilisera les variables d'environnement.

Exercice 3 : Le mot le plus long

Question 1

Dans cet exercice on s'intéresse à la recherche du mot le plus long d'un texte. Pour simplifier l'écriture d'un tel mécanisme, on va commencer par développer un script `longest_param.sh` affichant le plus long de ses paramètres (le premier en cas d'égalité).

Question 2

Dans un deuxième temps, utilisez votre script `longest_param.sh` pour implémenter un script `longest_word.sh` qui affiche le mot le plus long de chaque ligne saisie sur la console.

Vous allez pour cela utiliser la commande `read` qui prend en paramètre une liste de noms de variables. Cette commande va lire une ligne sur son *entrée standard*, puis la découper en éléments suivant la variable `$IFS` (par défaut espace, tabulation et retour chariot). Les $n-1$ premiers éléments sont affectés aux $n-1$ premières variables et le reste de la ligne est affecté à la dernière variable.

Cette commande retournant 0 en cas de succès, il est classique de l'utiliser de la façon suivante :

```
while read mot_1 mot_2 reste_des_mots ; do
  # quelques commandes utilisant ces trois variables
done
```

Question 3

On veut maintenant chercher les mots les plus longs dans un fichier et non sur la saisie au clavier, sans modifier le script `longest_word.sh`.

On décide donc de lancer la commande suivante, où `proverbe.txt` est un fichier contenant un adage sur une seule ligne. Que se passe-t-il ?

```
moi@pc /home/moi $ ./longest_word.sh proverbe.txt
```

Question 4

Pour arriver à nos fins, il faut arriver à forcer `longest_word.sh` à lire dans fichier `proverbe.txt` pour en afficher son mot le plus long.

Proposez une solution, qui utilise l'opérateur de redirection du flux d'*entrée standard* `<`, comme dans l'exemple suivant :

```
moi@pc /home/moi $ commande < fichier
```

Question 5

En *Bash* le mot clé `while` est une commande comme les autres. On peut donc aussi rediriger son flux d'*entrée standard*. Proposez une nouvelle version de `longest_word.sh` qui prenne en paramètre le nom d'un fichier (ne contenant qu'une ligne) (comme dans la question 3) et affiche son mot le plus long.

Question 6

On veut maintenant chercher le plus long mot du texte contenant plus d'une ligne. L'idée est de construire une chaîne contenant les plus longs mots de chaque ligne et de réutiliser le script `longest_param.sh` pour obtenir le mot recherché.

Pourquoi ne peut-on pas utiliser la valeur de retour avec `$?` pour construire une liste des plus longs mots de chaque ligne.

Question 7

La solution va reposer ici sur l'utilisation de la substitution de commandes. Son principe est le suivant. On place une ou plusieurs commandes au milieu de `$( )`. Ces commandes sont alors exécutées, puis l'ensemble est remplacé par les affichages de ces commandes.

Voici un exemple avec la commande `whoami` qui affiche le nom de celui qui la lance :

```
moi@pc /home/moi $ x=$(whoami)
moi@pc /home/moi $ echo "Mon␣login␣a␣${#x}␣lettres."
Mon login a 8 lettres.
```

Utilisez la substitution de commandes pour implémenter une nouvelle version de `longest_word.sh` qui affiche, en utilisant l'algorithme décrit dans la question précédente, le mot le plus long du texte.