TD3 : Les scripts shell

OBJECTIFS

- ✓ script shell et les variables d'environnement
- ✓ Acquérir des éléments de base de la programmation pour contrôler l'exécution des commandes par un interpréteur de commande.

1 VARIABLE D'ENVIRONNEMENT

La configuration de bash se trouve à différents emplacements.

Au niveau du système, deux fichiers de configuration :

`/etc/profile` (chargé juste après s'être loggué)

`/etc/.bashrc` (chargé pour une session déjà logguée. Exemple : lors d'une session graphique, les terminaux virtuels s'ouvrent sans demander de s'authentifier car l'utilisateur s'est déjà loggué).

Ces fichiers de configuration ne sont modifiables que par le root et affectent tous les utilisateurs du système.

Il est possible d'appliquer une configuration par utilisateur :

`~/.bash_profile` (chargé juste après s'être loggué)

`~/.bashrc` (chargé pour une session déjà logguée)

Au démarrage du shell, Bash va automatiquement charger la configuration système puis ensuite la configuration propre à l'utilisateur en cours.

Variables en bash

Exercice 1 :

1. Afficher toutes les variables d'environnement du système. Et compter les.
2. Afficher le contenu de la variable d'environnement « PATH ».
3. Afficher le nom de l'utilisateur courant suivi de son répertoire personnel.
4. Définir une variable locale: annee=2021.
5. Ouvrir une nouvelle session shell (taper la commande sh)
6. Taper la commande echo $annee. Que constatez-vous ?
7. Exporter la variable annee et refaire 6.
8. Rajouter cette variable au fichier .bashrc et refaire 6.

2 SCRIPT D'AUTOMATISATION DE TACHE

Sous Linux, « **bash** » est l'interpréteur de commande de base chargé d'exécuter ces instructions de commande. Les commandes sont habituellement éditées sur une seule ligne par commande. Il est possible de traiter plusieurs commandes en une seule ligne en les séparant par un point-virgule. Elles sont exécutées séquentiellement (l'une après l'autre)

Structure script élémentaire

Linux permet de regrouper plusieurs instructions dans un fichier texte ayant des droits d'exécution pour lancer automatique leur exécution à son invocation dans une console. On parle de « **script** ».

*Il est possible d'exécuter des scripts implémentés dans d'autre langage et de les exécuter en utilisant l'interpréteur de commande adapté. Vous devez l'invoquer dans la console en précisant par **son chemin d'accès** (chemin relatif ou abolu)*

*Pour exécuter le script automatique, il faut indiquer au système d'exploitation que le fichier est un script exécutable. Inscrivez en première ligne du fichier une instruction débutant par « **#!** » suivi de l'interpréteur de commande à utiliser avec son chemin d'accès – cette première instruction s'appelle le « **shebang** » .*

***Exemples** : Shell **bash** → `#!/bin/bash`*
*Interpréteur **perl** → `#!/usr/bin/perl`*

***Prog1** : A la racine de votre dossier d'accueil, créez un dossier « ~/myscripts » et placez-vous à l'intérieur. En utilisant les variables d'environnement existantes, créez un premier script « ~/myscripts/bienvenue.sh » qui donnera lors de son exécution l'affichage suivant :*

```
Bienvenue xxxx
Vous êtes sur le poste yyyy
Votre dossier d'accueil est /zzzz
```

xxxx : *correspond au nom du compte utilisateur*

yyyy : *correspond au nom de la machine*

/zzzz : *correspond au chemin absolu du répertoire d'accueil de l'utilisateur*

Modifier les droits sur le fichier pour ajouter le droit d'exécution au propriétaire, au groupe et et tous les utilisateurs. Testez le dans une console pour vérifier la juste interprétation des variables d'environnement.

Passage d'argument à un script

On peut transmettre à un script des arguments ou paramètres nécessaires à son exécution directement sur la ligne de commande.

Les paramètres de position « **$1** », « **$2** », « **$3** », …, renvoient à la valeur des paramètres (ou arguments) passés au script conformément à leur ordre de transmission. La numérotation des paramètres de position démarre à la valeur « 1 », le paramètre « **$0** » est considéré comme un paramètre spécial – voir ci-dessous.

*Le **shell Bourne** est limité aux paramètres de 0 à 9. Si le nombre de paramètre est supérieur à 9, on récupère leur valeur en l'encapsulant dans des accolades – exemple « **${10}** », « **${11}** », …*

Les paramètres spéciaux sont des variables réservées permettant d'effectuer des traitements sur les paramètres eux-mêmes.

« **$0** » → Premier argument de la ligne de commande, correspond au nom du script invoqué

« **$*** » → Tous les arguments de position regroupés en un seul

« **$@** » → Tableau de tous les paramètres de position

« **$#** » → Nombre de paramètres de position

```
#!/bin/bash

echo "Le nombre d'argument(s) passé(s) est $#"

echo "Le premier argument est $1"

echo "Le dernier argument est $#"
```

Exercice 2 :

Ecrire un script shell qui permet de changer l'extension des fichiers .cpp qui se trouvent dans un répertoire passé en argument par une deuxième extension passé en argument aussi.

Exercice 3 :

1. Créer la commande **crefic1** obéissant à la syntaxe suivante :

crefic1 nomfichier
permettant de créer un fichier nomfichier dans le répertoire courant.

2. Créer la commande **crefic2** obéissant à la syntaxe suivante :

crefic2 nomfichier
permettant de créer un fichiers nomfichier dans le répertoire courant après validation de l'utilisateur:
$./crefic2 toto
voulez vous créer toto ?
Oui
toto est crée
$./crefic2 test
voulez vous créer test?
non

3. Créer la commande **crefic3** obéissant à la syntaxe suivante :

crefic3 nom N
Son rôle est de créer un ensemble de fichiers nom1, nom2, …nomN. La création de chaque fichier doit être validée en interactif par l'utilisateur.

Exercice 4 :

Ecrire un script nommé indice qui affiche l'indice de son premier paramètre dans la liste des paramètres qui suivent.

Exemples :

```
$ indice toto tata titi toto tutu

L'indice de toto dans la liste est : 3
```

```
$ indice 9 2 8 6 9 5 0

L'indice de 9 dans la liste est : 4
```

Exercice 5 :

Ecrire un script qui affiche le menu suivant, et ensuite demande à l'utilisateur de rentrer un nombre (de 1 à 4), et qui affiche le résultat de l'exécution de la commande entre les parenthèses selon l'entrée de l'utilisateur :

1. Lister le contenu du répertoire ($ls -l)
2. Lister les processus en cours d'exécution ($ps aux)
3. La date d'aujourd'hui ($date)
4. Quitter

Exercice 6 :

Ecrire un script qui prend comme argument un fichier et transforme toutes les lettres minuscules en majuscules en dans son contenu.

Exercice 7 :

Ecrire un script "test-fichier.sh", qui précisera le type du fichier passé en argument, ses permissions d'accès pour l'utilisateur. Si le script ne reçoit aucun paramètre, il devra afficher un message d'erreur.
Exemple de résultats :
Le fichier /etc est un répertoire
"/etc" est accessible par root en lecture écriture exécution
Le fichier /etc/smb.conf est un fichier ordinaire qui n'est pas vide
"/etc/smb.conf" est accessible par toto en lecture.