TD07 -Les Scripts Shell

M1204 - Services sur Réseaux

Patrice Gommery - Novembre 2019

Pour continuer ce semestre, nous allons dans ce TD, revenir sur la partie administration du serveur avec l'utilisation des commandes de bases Linux et la création de scripts permettant d'automatiser certaines tâches d'administration de notre serveur.

CONSIGNES GENERALES :

Dans tout l'exercice,
remplacez **N** par votre numéro de table (*visible en ouvrant un terminal*)

Créez une machine virtuelle sous **VirtualBOX** en utilisant le fichier **debian.vdi** disponible sur votre Bureau (Linux Debian 64 Bits). N'oubliez pas de passer la machine en mode d'accès par pont pour accéder à toutes les serveurs de la salle réseau.

Modifiez l'adresse IP de la machine en 172.16.**N**.250
Installez le paquet **proftpd** nécessaire pour la suite du TD.
Installez aussi le paquet **ftp** pour pouvoir accéder au serveur FTP de la salle.

Rappel : Le **serveur FTP** de la salle se trouve à l'adresse : **172.16.100.1**,
Sur ce serveur, vous disposez d'un compte **mmis1** avec le mot de passe : **PASSWORD**

PARTIE-1 : RETOUR CHEZ LES GAULOIS

Pour illustrer ce TP, nous allons revenir dans notre village de gaulois préféré et voir comment les scripts shell pourraient nous aider. Pour commencer, un petit rappel de l'exercice :

- Créez 4 utilisateurs : **asterix**, **obelix**, **panoramix** et **cesar** (mots de passe 123 pour tous)
- Créez 2 groupes : **gaulois** et **romains**
- Répartissez les quatre utilisateurs dans les deux groupes (je vous laisse deviner lesquels)

- Dans la racine de l'arborescence, créez un dossier **village**
- Dans ce dossier créez deux fichiers vides : **potion.txt** et **sangliers.txt**

- Modifiez les droits d'accès au dossier et aux fichiers pour que :
  - **Obelix** puisse modifier **sangliers.txt**, mais n'accède pas à **potion.txt**
  - **Asterix** puisse modifier **sangliers.txt** et uniquement lire **potion.txt**
  - **Cesar** ne doit pas avoir accès au dossier **village** , ni à son contenu
  - **Panoramix** doit pouvoir accéder à tout avec tous les droits.

et sur les commandes mises en oeuvre : **`adduser, groupadd, usermod, mkdir, touch, chown, chmod, ls`** et **`cd`**. Vous ne vous rappelez pas des syntaxes ? Utilisez **`man`** ou reprenez le CM sur les commandes Linux (et vos notes ?)

Vous avez tout recréé ? Bravo, mais inutile si vous aviez lu le TP avant de commencer à saisir les commandes. En effet , l'enseignant à créer un script pour faire une correction rapide. Celui-ci est disponible sur le serveur FTP de la salle (*Voir consignes générales*).

Le script se nomme **`village.sh`** et se trouve dans le dossier **`/mmis1/scripts/`**
de ce serveur .

Téléchargez ce script dans votre dossier **`/root`** .
Rendez le exécutable avec la commande **`chmod 750 /root/village.sh`**
et exécutez-le : **`/root/village.sh`**

Le script va automatiquement créer le dossier **/village**, mais aussi tous les utilisateurs et les groupes avec les bons droits sur les bons dossiers et fichiers. Vérifiez par vous-même avec les commandes :

```
ls -l / | grep village
ls -l /village
```

Les résultats retournés devraient-être ceux-là :

```
d rwx r-x ---  panoramix gaulois   ...  /village

- rwx rwx ---  panoramix gaulois   ...  /village/sangliers.txt
- rwx r-x ---  panoramix potion    ...  /village/potion.txt
```

J'espère que ceux qui ont tout recréé en tapant les commandes auront compris la leçon.
IL FAUT LIRE LE TP AVANT DE SAISIR BETEMENT LES COMMANDES :-)

PARTIE-2 : LES NOUVEAUX GAULOIS

Bien, votre village est bien en place, mais que se passe-t-il si de nouveaux gaulois arrivent ? Il faut bien sur reprendre les mêmes opérations que pour Asterix et Obélix :

- Créer l'utilisateur (adduser)
- Le mettre dans les groupes Gaulois et Potion (usermod -aG)
- Faire du Village son dossier personnel (usermod -d)

Les opérations sont donc très répétitives et seul le Nom et le mot de Passe est différent pour chacun des utilisateurs. Une automatisation des opérations permettrait donc de gagner un temps précieux et de ne pas oublier une étape lors de la création d'un nouvel utilisateur. Comme vous l'avez vu dans la partie précédente, sous Linux, cette "automatisation" est possible grâce à l'utilisation de scripts.

Nous allons donc en utiliser un pour créer notre prochain gaulois.
Le script est aussi disponible sur le serveur FTP de la salle dans le dossier **/mmis1/scripts** et se nomme : **gaulois.sh**

Comme pour le script précédent, téléchargez le fichier gaulois.sh dans votre dossier **/root** et rendez-le exécutable.

Ensuite, nous pouvons simplement utiliser le script en tapant : **/root/gaulois.sh**

Le script vous demande alors le nom du gaulois que vous désirez créer et exécute toutes les opérations nécessaires à sa création. Vous pouvez vérifier que tout est bien créé en vous connectant avec Filezilla et le nom du gaulois comme compte utilisateur .

*REMARQUE : Pour se connecter en FTP sur votre serveur, il faut bien sûr que le package **proftpd** est été installé au début du TD. Si ce n'est pas le cas , installez-le et configurez-le en modifiant le fichier* **/etc/proftpd/proftpd.conf** *(Décommentez la ligne # Default Root ~)*

Vous pouvez exécuter le script autant de fois que vous le désirez.

PARTIE-3 : INTRODUCTION AUX SCRIPTS SHELL

Comme vous avez-pu le constater, l'utilisation de scripts permet d'automatiser et de simplifier l'administration Linux de façon considérable, mais comment sont écrits ces scripts ? Tout simplement avec un langage propre au système Linux et plus particulièrement à son interface de commande, qui est dans notre cas : **Le Shell Bash**.

Le Shell Bash fournit donc à la fois l'interface qui nous permet de taper les commandes systèmes , mais aussi un langage de programmation complet, capable d'interaction avec l'utilisateur, mais aussi d'effectuer des tests de condition et des boucles comme peut le faire PHP que vous utilisez dans le domaine du Web.

D'ailleurs, comme PHP, le Shell Bash est un langage interprété, les commandes sont donc exécutées directement, les unes après les autres, pas besoin de compiler les instructions.

Un script est directement opérationnel après son écriture, **une seule condition : qu'il soit déclaré exécutable pour l'utilisateur qui voudrait l'utiliser.** C'est ce que vous avez fait avec la commande **chmod** après avoir téléchargé les scripts du TP.

Comment exécuter un script ? Il suffit de l'appeler en précisant son chemin d'accès. Dans notre exemple, les scripts se trouvant dans le dossier **/root**, nous avons tout simplement saisi : **/root/NomDuScript** , nous aurions aussi pu nous placer dans le dossier **root** et taper **./NomDuSript** (le **.** désignant le répertoire en cours)

Comment nommer les scripts ? Comme vous voulez, les extensions n'ayant aucune importance pour notre système Linux. Par convention , on "suffixe" un script S**hell** par **.sh**, mais nous aurions pu aussi bien mettre **.com** , **.exe**, **.cmd** ou même rien du tout .

Comment créer ou modifier un script ? Tout simplement avec un éditeur de texte. Les scripts sont de simples fichiers textes. Dans notre cas nous utiliserons **nano** qui suffit amplement à nos besoins, mais il existe des éditeurs plus sophistiqués comme **vim** ou **emacs** qui fournissent des fonctions plus élaborées. Vous pouvez aussi écrire vos scripts sur une autre machine avec des éditeurs dédiés au développement comme **Sublime Text** ou **Scite** qui sont présents sur votre poste de travail. Vous profiterez alors de la coloration syntaxique de ces éditeurs.

Pour la suite du TP, je vous propose d'ouvrir le script **gaulois.sh** que vous avez téléchargé sur votre serveur et de regarder quelques-unes des instructions qui le composent.

REMARQUE : Vous pouvez aussi le télécharger avec **Filezilla** sur votre poste de travail et l'ouvrir avec **Sublime** ou **Scite**.

ENTETE DU FICHIER :

```
#!/bin/bash
#Script de Création d'un nouveau Gaulois
#Usage : ./gaulois.sh
```

Tout fichier script commence par la déclaration du Shell utilisé. I existe d'autres shells comme ksh ,sh, bsh etc .. . Ces Shells sont très peu différents de bash que nous utilisons, mais peuvent utiliser d'autres syntaxes ou d'autres commandes qui ne seraient alors pas interprétées par notre système. Tous les systèmes Linux n'utilisent pas non plus le même Shell par défaut.

Notre fichier commence donc obligatoirement par : **#!/bin/bash**
Les lignes qui suivent et qui commencent par **#** ne sont que des **commentaires**.

VARIABLES :

Comme tout langage de programmation, Bash sait utiliser et manipuler des variables.
Pour les déclarer, rien de plus simple : **NomVariable=Valeur**
Pour les utiliser, il suffit de les appeler avec **$NomVariable**

```
dossier="/village"
echo "Le dossier personnel de votre Gaulois sera  : "$dossier
```

Ici, on déclare la variable **dossier** avec pour valeur **/village** . La valeur entre double-cotes permet de préciser que la variable contient du texte (type **string**).
Ensuite, on l'utilise en l'appelant avec **$dossier**
La commande **echo** affiche le contenu de la variable **dossier**, donc **/village** précédée du texte .

REMARQUE : Vous pouvez taper ces deux commandes directement à l'invite de commandes et voir immédiatement le résultat.

TEST DE CONDITION :

```
if [[ -d $dossier ]]
then
     echo "le dossier $dossier existe"
else
     echo "le dossier $dossier n'existe pas, impossible de créer un Gaulois"
     exit
fi
```

C'est le classique IF présent dans tous les langages de programmation et qui permet de tester le contenu d'une variable, mais aussi la présence d'un fichier ou d'un dossier.

Ici, nous testons la présence du dossier **/village** .
S'il n'existe pas, on sort du script avec la commande **exit**

La syntaxe est la suivante :

if [[$NomVariable Test de Condition]] - *les* [*et les* ESPACES *sont importants.*

```
then
.... Code si la condition est vraie
else
.... Code si la condition est fausse
fi
```

Pour tester la présence d'un fichier : **if [[-f Chemin/NomFichier]]**
Pour tester la présence d'un dossier : **if [[-d Chemin/NomDossier]]**

BOUCLES :

```
while [[ $gaulois == '' ]]
do
    echo "Comment se nomme votre nouveau Gaulois ?"
    read gaulois
done
```

Ici une boucle **WHILE** qui s'exécutera donc tant que la condition sera remplie.
Dans notre cas, on teste si la variable gaulois est vide, si c'est le cas on exécute la commande **read** qui permet de demander une valeur à l'utilisateur.

La syntaxe est la suivante :

```
while [[ Condition ]]
do
     .... Code à répéter tant que la condition est vraie ...
done
```

Il existe bien sur d'autres instructions permettant de réaliser des boucles comme par exemple le classique FOR qui permet de boucler sur une séquence définie.

La boucle suivante, affiche les valeurs de 1 à 10

```
for i in `seq 1 10`;
do
        echo $i
done
```

Celle-ci affiche les valeurs consécutives attribuée à la variable légumes :

```
for legume in 'tomate' 'oignons' 'poireaux'
do
        echo "Le légume du jour est  $legume"
done
```

Celle-ci affiche tous les fichiers du dossier courant

```
for fichier in `ls`
do
        echo "Fichier trouvé : $fichier"
done
```

COMMANDES D'AFFICHAGE :

Nous l'avons plus haut, comme en PHP, il existe une commande **echo** qui permet donc d'afficher du texte ou le contenu d'une variable. Mais elle permet aussi ,associée à certains opérateurs , de réaliser d'autres actions. Par exemple dans notre script elle est utilisée avec **les opérateurs de redirection >** et **>>** pour créer et compléter des fichiers.

Comment fonctionnent ces opérateurs ? Ils redirigent tout simplement la sortie de la commande (qui normalement est l'écran) vers le fichier qui suit l'opérateur.

L'opérateur **>** seul **créait** un nouveau fichier et **écrit** le flux dans celui-ci.
L'opérateur **>>** (Deux symbole supérieurs) **écrit** simplement le flux à la **fin** du fichier.

L'autre commande d'affichage est **printf**. Elle se différencie de **echo** par le fait qu'elle accepte des arguments et des options permettant de formater le texte à afficher. Par exemple dans notre script le caractère **\n** en fin de ligne permet de faire un saut de ligne.
La commande **man printf** vous donnera plus d'options sur cette commande.

INTERACTIVITE :

Le langage permet aussi de demander des informations à l'utilisateur grâce à la commande **read** qui stockera la saisie de l'utilisateur dans une variable.

Exemples dans notre script :

```
echo "Comment se nomme votre nouveau Gaulois ? :"
read  gaulois
```

Ici la commande **echo** affiche le texte de la demande, la commande **read** attend la saisie de l'utilisateur et la stocke dans la variable **gaulois**.

Autre exemple avec des options :

```
read –s –n1 –p "Appuyez sur une touche pour continuer..."; echo
```

L'utilisation de l'option **-p** permet de passer directement le message à afficher avec la commande **read**.
L'option **-s** empêche de voir la saisie (comme pour les mots de passe).
L'option **-n1** limite la lecture a 1 caractère. Ici aucune donnée n'est enregistrée.
On attend juste que l'utilisateur appuie sur une touche pour passer à l'instruction suivante.

LES AUTRES COMMANDES :

Les autres commandes sont des commandes classiques du système, comme :
- **useradd** : création d'un utilisateur
- **usermod** : paramétrage de l'utilisateur
- **chpasswd** : modifie le mot de passe de l'utilisateur

PETITE EXPLICATION SUR L'UTILISATION DE LA COMMANDE chpasswd

La syntaxe utilisée dans le script est la suivante : **echo "$gaulois:123" | chpasswd**

- La commande **chpasswd** lit une liste de paires de noms d'utilisateurs et de mots de passe depuis l'entrée standard et utilise ces informations pour mettre à jour un groupe d'utilisateurs existants. Chaque ligne est au format suivant : nom_utilisateur:mot_de_passe

- Le "**pipe**" (symbole |) permet de rediriger la sortie standard vers une autre commande.

Ici, nous renvoyons donc l'affichage du echo *(nom:password)* **vers la commande chpasswd.**

Toutes les commandes du système sont ainsi manipulables par un script et l'on peut donc ainsi automatiser un maximum de choses pour administrer le système comme on le désire. Regardez le contenu du script **`village.sh`** , vous constaterez qu'il ne contient qu'une suite de commandes Linux.

Les instructions utilisées dans ces scripts ne sont qu'une toute petite partie de ce que l'on peut réaliser avec des scripts bien construits. Pour en savoir plus :

https://guidespratiques.traduc.org/guides/vf/Bash-Beginners-Guide/Bash-Beginners-Guide.html#sect_07_01

EXERCICE :

Pour terminer ce TP, sur votre machine dans le dossier **/root**, créez un script nommé **info.sh** qui demande à l'utilisateur son **nom**, son **prénom,** son **vmid**
Le script devra écrire ces informations dans un fichier nommé **vmid.txt**
Attention, prenez en compte le fait que l'on puisse exécuté plusieurs fois le script,
il faudra donc testé l'existence ou pas du fichier.
Le fichier ne doit pas être écrasé s'il existe déjà.