Mécanismes essentiels du Shell GNU/Linux

Compte Rendu : votre CR contiendra les réponses aux questions posées à déposer avant la fin de la séance dans le répertoire à votre nom du dépôt AdminSystem. Nom du fichier : **05-MécanismesEssentielsDuShell**

1. Le shell bash – Généralités

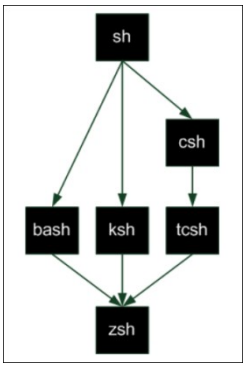
1.1. Présentation

Il est inenvisageable pour un professionnel de Linux de ne pas connaître le fonctionnement de l'interpréteur de commandes et des principales commandes qui lui sont associées !! **L'interpréteur de commandes est un langage appelé shell.**

C'est à rapprocher du mot kernel : le kernel, signifiant noyau, est souvent entouré d'une coquille dure (comme un noyau d'abricot ou de pêche). Shell signifiant coquille, c'est donc ce qui « entoure » le noyau Linux : le moyen de l'utiliser à l'aide de commandes.

Le langage shell permet d'automatiser la plupart des tâches. Ce langage est totalement intégré à GNU/Linux et c'est un langage interprété : il n'y a rien à compiler.

Il existe plusieurs shells, chacun disposant de spécificités propres. Le Bourne Shell (sh) est le shell le plus connu et le plus courant sur les Unix. Il existe plusieurs shells, voici les principaux :



- **sh** : *Bourne Shell*. L'ancêtre de tous les shells.
- **bash** : *Bourne Again Shell*. Une amélioration du *Bourne Shell*, disponible par défaut sous Linux et Mac OS X.
- **ksh** : *Korn Shell*. Un shell puissant assez présent sur les Unix propriétaires, mais aussi disponible en version libre, compatible avec bash.
- **csh** : *C Shell*. Un shell utilisant une syntaxe proche du langage C.
- **tcsh** : *Tenex C Shell*. Amélioration du *C Shell*.
- **zsh** : *Z Shell*. Shell assez récent reprenant les meilleures idées de bash, ksh et tcsh.

Nous utilisons le « bash » qui est activé par défaut pour les utilisateurs Debian, on le trouve par défaut sous Linux et Mac OS X.

La liste des shells actuellement présents sur votre installation Linux est présente dans le fichier /etc/shells.

➢ Consultez ce fichier et citez les shells installés (commande « **cat nomFichier** » pour afficher sans modifier)

1.2. Bash : le shell par défaut

Le bash est un dérivé du **Bourne Shell**. Bourne est le nom du principal programmeur de ce shell. L'expression Bourne Again est à la fois un clin d'oeil aux origines du bash (Bourne), et un jeu de mots sur « **I born again** » ce qui signifie « né de nouveau » ou « réincarné ». Le bash reprend sh mais aussi des fonctionnalités de ksh ou csh.

2. Rappels sur les redirections et tubes

Rappel : l'aide sur une commande Linux est accessible en tapant *man <commande>*.

Etudions la commande ***wc*** (qui signifie word count):

wc [-options] fic
-c : donne le nombre de caractères
-w : donne le nombre de mots
-l : donne le nombre de lignes

- Testez cette commande sans paramètre sur le fichier « .bashrc ». Que signifient les 3 nombres ?

- Connectez-vous directement sur la machine Debian (donc sans passer par putty) afin d'avoir une 2ème connexion (faites l'inverse si vous êtes déjà connecté directement sur la MV).
- Testez la commande suivante (pour bien comprendre, il est nécessaire de tester la 1ère partie de la commande seule) :

$ who | wc -l

Cette suite de commandes permet d'obtenir le nombre d'utilisateurs connectés. On constate que la combinaison de deux commandes simples permet d'aboutir à un résultat intéressant. → Déconnectez-vous de la session sur Debian et relancez cette commande.

Expliquez ce que vous avez compris sur cette suite de commandes **who | wc -l**

- Testez la commande suivante (même remarque, testez la 1ère partie avant le | à part) :

$ cat /etc/passwd | wc -l

Que fait cette commande (le fichier passwd contenant les utilisateurs du système) ?

- Testez la commande suivante :

$ cat > fich1

Coucou

Hello

Bonjour

Ctrl-D

$ cat fich1

- **Que contient le fichier fich1 ?**

- Testez la commande suivante :

$ echo "Guten tag" >> fich1

- Vérifiez le contenu du fichier fich1.

$ echo "Here comes the sun" > fich1

- Vérifiez le contenu du fichier fich1.
- **Pourquoi ce changement ?**

$ echo "Fly me to the moon" > fich2

$ echo "House of the rising sun" >> fich3

- Vérifiez le contenu des fichiers fich2 et fich3.

$ cat fich1 fich2 fich3 > res

- **A votre avis, sans regarder, que doit contenir le fichier res ? Vérifiez.**

3. Commandes système

Rappel : l'aide sur une commande Linux est accessible en tapant *man <commande>*.

Trouvez les informations affichées par les commandes ci-dessous.

uname -a

who

pwd

df # Comment afficher le résultat de cette commande en Mo, Go, … ?

ip route

hostname

Que font les commandes ci-dessous ? (Attention, certaines nécessitent des paramètres)

clear

touch

more

halt

4. Recherche de chaîne de caractères

Rappel : l'aide sur une commande Linux est accessible en tapant *man <commande>*.

Etudions à présent la commande ***grep***

grep [-options] modèle_de_critères [fichier1 fichier2 .....]

-c donne seulement le nombre de lignes trouvées obéissant au critère
-l donne seulement le nom des fichiers où le critère a été trouvé
-v donne les lignes où le critère n'a pas été trouvé
-i ne pas tenir compte de la casse (ne pas différencier majuscules/minuscules)
-n pour n'afficher que les numéros des lignes trouvées

-w pour imposer que le motif corresponde à un mot entier d'une ligne

Le modèle de critère (ou **motif**) peut utiliser des caractères spéciaux, en fait les mêmes que ceux utilisés par l'éditeur ***vi.***

Exemple : la commande ci-dessous liste l'utilisateur "root" parmi les utilisateurs connectés.

$ ps au | grep root

Ici, la commande ps liste les processus exécutés et sa sortie standard est dirigée vers la commande grep, qui filtre les lignes en recherchant dans la ligne la chaîne de caractère root.

Testez les commandes ci-dessous.

Testez les commandes ci-dessous.

$ grep backup /etc/passwd

$ grep -v backup /etc/passwd

Expliquez la différence entre ces deux commandes

- Le caractère ^ placé devant les arguments signifie que le modèle recherché doit se situer en début de ligne.
- Le caractère **$** placé derrière les arguments signifie que le modèle recherché doit se situer en fin de ligne.

Il s'agit d'expressions régulières dont le détail (important) ne rentre pas dans ce cours.

$ cd

$ ls -l | grep '^d'

Que renvoie cette commande ?

Voici une liste de symboles utilisables par grep : . * [ ] [^] ^ $

- . signifie un caractère quelconque
- * répétition du caractère situé devant
- ^ début de ligne
- $ fin d'une ligne (donc "e$" mots se terminant par e)
- [...] contient une liste ou un intervalle de caractères cherchés
- [^..] caractères interdits.

Pour éviter une confusion entre les interprétations de ces symboles spéciaux par grep ou par le shell, il est indispensable de "verrouiller" le motif en plaçant l'expression entre guillemets " " (et non entre quotes !).

Vous allez créer un fichier test.txt qui contient une liste des fichiers et répertoires de toute l'arborescence de votre système Debian. Exécutez la commande suivante pour cela :

$ ls -lR / > test.txt

Cette commande permet de lister tous les fichiers et répertoires de manière récursive à partir de la racine (commande « ls -lR / ») et de rediriger le résultat (opérateur « > ») dans le fichier test.txt.

Combien de lignes contient le fichier test.txt ?

Tester et expliquer les commandes suivantes avec le fichier test.txt construit précédemment.

<u>Rappel</u> : l'aide sur une commande Linux est accessible en tapant *man <commande>*.

$ grep "sun" test.txt

$ grep -c "sun" test.txt

$ grep -v "sun" test.txt

$ grep "sun$" test.txt

5. Recherche de fichiers

Rappel : l'aide sur une commande Linux est accessible en tapant *man <commande>*.

La commande find recherche des fichiers dans l'arborescence et peut exécuter une action sur ces fichiers. L'arbre parcouru est celui désigné par le premier argument.

La commande ci-dessous recherche tous les sous-répertoires du répertoire courant.

$ find . -type d

Que désigne le caractère point (.) ?

Testez et expliquez les commandes suivantes.

$ find . -user root (ou tapez le nom de l'utilisateur connecté)

$ find . -type f -size +10k

6. Groupement de commandes

Rappel : l'aide sur une commande Linux est accessible en tapant *man <commande>*.

Le chaînage de commande est possible avec **« ; »**. Il est aussi possible de grouper les commandes.

- Que se passe t'il quand vous exécutez les commandes « **uname –a ; pwd ; ls –l >resultat.txt** » ?

Pour comprendre cette ligne de commande, commencez par tester chaque commande unitairement. Pour voir le contenu du fichier resultat.txt, utilisez la commande « cat resultat.txt ».

- Quelle est la différence entre la suite de commandes précédentes et « **uname –a >resultat.txt ; pwd >>resultat.txt ; ls –l >>resultat.txt** » ?

- Quelle est la différence avec « **{ uname –a ; pwd ; ls –l ; } > resultat.txt** » ?

7. Les archives

Rappel : l'aide sur une commande Linux est accessible en tapant *man <commande>*.

Effectuez les actions ci-dessous dans l'ordre puis **complétez chaque cellule vide du tableau.**

Commande	Résultat
cd	
ls -la	Affichage du contenu du répertoire
tar cfv moi.tar *	Archivage dans le fichier moi.tar du contenu du répertoire
ls -la	Voir le résultat de la commande précédente
tar tfv moi.tar	
gzip moi.tar	Compression de l'archive (création du fichier moi.tar.gz)
ls -la	Voir le résultat de la commande précédente
mkdir /tmp/test/	Création du répertoire /tmp/test
mv moi.tar.gz /tmp/test	
	Déplacement dans le répertoire /tmp/test
tar zxfv moi.tar.gz	
	Déplacement dans le répertoire /tmp/
rm -Rf test/	

8. Utilisateurs et groupes

Rappel : l'aide sur une commande Linux est accessible en tapant *man <commande>*.

Consultez l'aide sur les commandes useradd et userdel et trouvez les commandes qui effectuent les actions suivantes (à faire avec les droits d'admin).

Créer l'utilisateur Toto en créant son répertoire personnel avec "Toto" en mot de passe. Quelle est la commande que vous avez utilisée ?

useradd -m Toto -p Toto

Vérifier dans le fichier /etc/passwd que cet utilisateur a été créé. Quel est son identifiant ?

Vérifier que son home répertoire a été créé dans /home. Qui est propriétaire du répertoire /home/Toto ?

Se connecter en tant que Toto. Quelle est la commande ?

Pouvez-vous effectuer la commande "ls -l" ? Pourquoi ? (Pour vous aider, taper la commande "pwd")

Que faire pour pouvoir exécuter cette commande "ls -l" ?

Revenir à l'utilisateur précédent (root normalement)

Supprimer Toto

Son répertoire personnel dans /home a-t-il été supprimé ? l'utilisateur dans /etc/passwd est-il supprimé ?

Quelle commande faut-il exécuter pour supprimer également le répertoire personnel ?

9. Les variables

On en distingue principalement deux types : variables utilisateur et variables système.

Le principe est de pouvoir affecter un contenu à un nom de variable, généralement une chaîne de caractère ou des valeurs numériques.

9.1. Nomenclature

Un nom de variable obéit à certaines règles :

- Il peut être composé de lettres minuscules, majuscules, de chiffres, de caractères de soulignement.
- Le premier caractère ne peut pas être un chiffre.
- Le taille d'un nom est en principe illimitée (il ne faut pas abuser non plus).
- Les conventions veulent que **les variables utilisateur soient en minuscules** pour les différencier des **variables système qui sont en majuscules**.

9.2. Quelques variables système

Le shell est lancé avec un certain nombre de variables prédéfinies utiles pour certaines commandes et accessibles par l'utilisateur. Le contenu de ces variables système peut être **modifié** mais il faut alors faire attention car certaines ont une **incidence directe sur le comportement du système**.

Pour consulter le contenu d'une variable, il faut utiliser la commande « **echo** » et faire précéder le nom de la variable du caractère « **$** ». Par exemple « **echo $HOME** ». La complétion automatique est utilisable avec les variables.

Donnez le contenu des variables suivantes :

Variable	Description	Contenu
HOME	Chemin d'accès du répertoire utilisateur. Répertoire par défaut en cas d'utilisation de CD.	
PATH	Liste de répertoires, séparés par des « : », où le shell va rechercher les commandes externes et autres scripts et binaires. La recherche se fait dans l'ordre des répertoires saisis.	
PS1	Prompt String 1, chaîne représentant le prompt standard affiché à l'écran par le shell en attente de saisie de commande.	
PS2	Prompt String 2, chaîne représentant un prompt secondaire au cas où la saisie doit être complétée.	
HISTFILE	Nom du fichier historique, généralement $HOME/.sh_history.	
HISTSIZE	Taille en nombre de lignes de l'historique.	
RANDOM	Génère et contient un nombre aléatoire entre 0 et 32767.	

9.3. Déclaration, affectation et affichage des variables utilisateur

Une variable est déclarée dès qu'une valeur lui est affectée. **L'affectation est effectuée avec le signe =, sans espace avant ou après le signe**.

Vous accédez au contenu d'une variable en plaçant le signe **$** devant le nom de la variable. Quand le shell rencontre le **$**, il tente d'interpréter le mot suivant comme étant une variable. Si elle existe, alors le $nom_variable est remplacé par son contenu, ou par un texte vide dans le cas contraire.

➢ Testez et expliquez chaque ligne de commandes :

Commande	Explication

var=Bonjour	Affectation d'une valeur à la variable $var
echo var	
echo $var	
chemin=/etc/ssh/	
l $chemin	
cd $chemin	

9.4. Un premier script

Editez avec **votre nouvel éditeur préféré** le fichier jeu.sh (**vi** jeu.sh) et saisissez exactement le contenu suivant (respectez les espaces, les indentations, la casse, les slashs et les anti slashs, … et vous devez obtenir les mêmes couleurs) :

```
#!/bin/bash
nbAtrouver=$RANDOM
gagne=0
while test $gagne -eq 0
do
    echo -e "\nProposer un nombre entre 0 et 32767"
    read nbPropose
    if [ $nbPropose -lt $nbAtrouver ] 2>/dev/null
    then
        echo -e "\a\t--> trop petit, recommencer"
    else
        if [ $nbPropose -gt $nbAtrouver ] 2>/dev/null
        then
            echo -e "\a\t--> trop grand, recommencer"
        else
            if [ $nbPropose -eq $nbAtrouver ]  2>/dev/null
            then
                echo -e "\t\n--> *** Bravo, c'est gagné, vous avez trouvé $nbAtrouver ***\n"
                gagne=1
            else
                echo -e "\a\t--> Erreur de saisie,recommencer"
            fi
        fi
    fi
done
~
~
```

Vous l'avez deviné, c'est un jeu qui propose de rechercher un nombre aléatoire. Pour exécuter ce fichier, il faut le rendre exécutable. La commande suivante permet de le rendre exécutable seulement pour l'utilisateur root, ce qui est suffisant.

```
root@Debian10Rev:~#
root@Debian10Rev:~# l jeu.sh
-rw-r--r-- 1 root root 578 sept. 10 17:19 jeu.sh
root@Debian10Rev:~#
root@Debian10Rev:~# chmod 744 jeu.sh
root@Debian10Rev:~#
root@Debian10Rev:~# l jeu.sh
-rwxr--r-- 1 root root 578 sept. 10 17:19 jeu.sh
root@Debian10Rev:~#
root@Debian10Rev:~#
```

Vous pouvez maintenant le lancer via la commande « ./jeu.sh » et l'utiliser.

Exercice : adaptez ce programme pour qu'il donne le nombre d'essai effectués avant de gagner comme cela :

```
Proposer un nombre entre 0 et 32767, essai n°9
32600
        --> trop grand, recommencer

Proposer un nombre entre 0 et 32767, essai n°10
32550
        --> trop petit, recommencer

Proposer un nombre entre 0 et 32767, essai n°11
32575

--> *** Bravo, c'est gagné, vous avez trouvé 32575 en 12 coups ***

root@Debian10Rev:~#
```

Et essayez de faire mieux 😉