SHELL

Qu’est-ce qu’un shell ?

Le shell a toujours été présent – à partir de l’aube d’UNIX.

Il s’agissait du programme qui a été exécuté après avoir entré votre nom de connexion et votre mot de passe sur le terminal pour accéder à l’ordinateur central.

Le premier programme Shell était le shell Thompson (SH) de 1971, qui a été remplacé en 1977 par Bourne Shell, également appelée SH. Tôt, il a été conçu de sorte que ce n’était qu’un autre programme pouvant être mis à jour facilement et que les utilisateurs pouvaient exécuter leur propre programme au lieu de la coque par défaut.

Le Shell est l’interpréteur en ligne de commande, il permet à un utilisateur de s’identifier sur une console ou un terminal.

Il fournit un certain nombre de commandes pour interagir avec le système d’exploitation.

En général, il permet aussi de créer des fichiers script que l’on nomme script pour exécuter plusieurs commandes l’une après l’autre. C’est donc aussi un langage de programmation.

JOB 1

La commande **ls** et ses options:

man ls : ls - Afficher le contenu de répertoires (man-linux-magique.net)

- Afficher le manuel de la commande ls

La commande: **ls** permet d'afficher le contenu du répertoire dans lequel on se situe (le contenu visible, excluant les fichiers cachés)

ls [OPTION]... [FICHIER]...

Afficher les informations des FICHIERS (du répertoire courant par défaut). Les entrées sont triées alphabétiquement si aucune des options

-cftuvSUX ou **--sort** n'est utilisée.

La commande **man** permet quant à elle d'expliciter la commande qu' elle précède, d'en afficher son manuel.

Pour afficher le manuel de la commande ls, on tapera donc la commande suivante:

man ls

- Afficher les fichiers cachés du home de votre utilisateur

Pour aller au-delà de l'affichage permis pas la commande **ls** , on s'aidera d'options affiliées à notre commande.

Pour afficher donc le contenu du répertoire en incluant les fichiers cachés qui s'y trouvent, on tape la commande: **ls -a** (**-a**, **--all** inclure les entrées débutant par « . »)

- Afficher les fichiers cachés plus les informations sur les droits sous forme de liste

Pour afficher les fichiers les fichiers cachés + les informations des droits sous forme de liste on ajoutera à notre commande ainsi qu'à notre option la suivante:
-l (utiliser un format d'affichage long)

On tape donc dans notre shell: **ls -a -l**

- Comment ajouter des options à une commande ?

Pour pouvoir ajouter des options à une commande Linux, il faut ajouter un "-" devant l'option que l'on veut. S'assurer également que l'option existe en jetant un oeil au manuel de la commande **(man)**

- Quelles sont les deux syntaxes principales d'écriture des options pour une commande ?

Les 2 syntaxes principales d'écritures des options pour une commande sont "-" et "--" précédant l'argument (ou l'option).

JOB 2

- Lisez un fichier en utilisant une commande qui permet seulement de lire

la commande **cat**
suivi du nom du fichier permet la lecture de son contenu seulement (sans pouvoir l'éditer)
cat .bashrc

- afficher les 10 premières lignes du fichier ".bashrc"

La commande **head** affiche par défaut les dix premières lignes d'un fichier texte.
head .bashrc

- afficher les 10 dernières lignes du fichier ".bashrc"

La commande **tail** permet de visionner par défaut les 10 dernières lignes d'un fichier texte.Cette outil est le plus souvent utilisé pour voir les fichiers log qui peuvent être très longs.
tail .bashrc

En ajoutant une option à la commande à ces commandes on peut alors spécifier le nombre de lignes que l'on souhaite faire apparaître.

- afficher les 20 premières lignes du fichier ".bashrc"

head -20 .bashrc

- afficher les 20 dernières lignes du fichier ".bashrc"

tail -20 .bashrc

JOB 3

La commande apt et ses options:

apt(8) — apt — Debian stretch — Debian Manpages

apt fournit une interface en ligne de commande pour le système de gestion de paquets. Elle est conçue comme une interface utilisateur et permet certaines options plus adaptées à une utilisation interactive par défaut par rapport aux outils d'APT plus spécialisés tels que apt-get(8) et apt-cache(8).

- Installer le paquet "cmatrix" **sudo apt-get install cmatrix**

- lancer le paquet que vous venez d'installer **cmatrix**

- Mettre à jour son gestionnaire de paquets **sudo apt-get update cmatrix**

- Mettre à jour ses différents logiciels **sudo apt-get upgrade cmatrix**

- Télécharger les internets : Google

Taper la commande suivante pour récupérer chrome depuis le web:

wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb

La commande suivante servira à installer notre fichier récupéré:
sudo dpkg -i google-chrome-stable_current_amd64.deb
Si des erreurs surviennent durant l'installation, corrigez les avec la commande suivante:
sudo apt-get install -f

- Redémarrer votre machine **sudo reboot**

- éteindre votre machine **sudo shutdown**

JOB 4

On commence par créer un fichier "users.txt"
touch users.txt
A l'aide de l'éditeur de text "nano", on modifiera le contenu de notre fichier (.txt)
nano users.txt
Pour y insérer le contenu suivant:
User1
User2
On veillera à sauvegarder les modifications effectuées avant de quitter (ctrl + x)

- Créer un groupe appelé "Plateformeurs"

Pour créer un nouveau groupe il faut taper la commande :
sudo groupadd Plateformeurs

- Créer un utilisateur appelé "User1"

Pour créer un nouvel utilisateur appelé "User1" il faut taper la commande
sudo useradd User1

- Créer un utilisateur appelé "User2"

Pour créer un nouvel utilisateur appelé "User2" il faut taper la commande
sudo useradd User2

- Ajouter "User2" au groupe Plateformeurs

Pour déplacer le nouvel utilisateur dans le nouveau groupe il faut taper:
sudo adduser User2 Plateformeurs

- Copier votre "users.txt" dans un fichier "droits.txt"

Pour copier notre "users.txt" dans un nouveau fichier nommé "droits.txt", nous utiliserons la commande **cp**.

cp users.txt droits.txt

- Copier votre "users.txt" dans un fichier "groupes.txt"

Même opération **cp users.txt groupes.txt**

- Changer le propriétaire du fichier "droits.txt" pour mettre "User1"

sudo chown User1 droits.txt

- Changer les droits du fichier "droits.txt" pour que "User2" ai accès seulement en lecture

Un utilisateur a le droit de faire un *chmod* sur un fichier :

- s'il est *root* ;
- ou s'il est le propriétaire du fichier en question.

```
rwxr-xr--
\ /\ /\ /
 v  v  v
 |  |  droits des autres utilisateurs (o)
 |  |
 |  droits des utilisateurs appartenant au groupe (g)
 |
droits du propriétaire (u)
```

Pour un fichier changer les droits d'un fichier

chmod [u g o a] [+ - =] [r w x] nom_du_fichier

Il est également possible d'utiliser les valeurs numériques au lieu d'alphabétique.

Correspondances de représentation des droits			
Droit	Valeur alphanumérique	Valeur octale	Valeur binaire
aucun droit	`---`	0	000
exécution seulement	`--x`	1	001
écriture seulement	`-w-`	2	010
écriture et exécution	`-wx`	3	011
lecture seulement	`r--`	4	100
lecture et exécution	`r-x`	5	101
lecture et écriture	`rw-`	6	110
tous les droits (lecture, écriture et exécution)	`rwx`	7	111

User1 est ici le propriétaire du fichier **droit.txt** , on souhaite accorder à **User2** un droit de lecture seulement à notre fichier. **User1** et **User2** ne faisant pas partie d'un même groupe, on modifiera donc les droits d'accès de **o** (other) en lui retirant les droits d'écriture et d'exécution.

Notre commande sera donc:

sudo chmod o-wx droits.txt ou **sudo chmod 774 droits.txt**

- Changer les droits du fichier "groupes.txt" pour que les utilisateurs puissent accéder au fichier en lecture uniquement

Dans la même logique (a signifiant all)

sudo chmod a-wx groupes.txt ou **sudo chmod 444 groupes.txt**

- Changer les droits du fichier pour que le groupe "Plateformeurs" puissent y

accéder en lecture/écriture.

Pour que le groupe **Plateformeurs** puisse avoir accès en lecture et écriture il nous faudra modifier le propriétaire groupe du fichier avec :

sudo chgrp Plateformeurs groupes.txt

On définit ensuite ses droits:

sudo chmod g+w

JOB 5

On ouvre le fichier **.bashrc,** grâce à l'éditeur de texte nano (ou un autre), dans lequel nous allons ajouter les alias suivants à la fin du fichier afin de les conserver et de pouvoir les réutiliser.

- Ajouter un alias qui permettra de lancer la commande "ls -la" en tapant "la"

alias la="ls -la"

- Ajouter un alias qui permettra de lancer la commande "apt-get update" en tapant"update"

alias update="apt-get update"

- Ajouter un alias qui permettra de lancer la commande "apt-get upgrade" en

tapant "upgrade"

alias upgrade="apt-get upgrade"

- Ajouter une variable d'environnement qui se nommera "USER" et qui sera égale à votre nom d'utilisateur

export USER=/home/rouissi

- Mettre à jour les modifications de votre bashrc dans votre shell actuel

Une fois les modifications apportées au fichiers .bashrc sauvegardées, quitter nano.
Taper la commande: **source .bashrc**
Pour recharger le bash et refléter les modifications apportées.
Une autre façon de recharger les modifications apportées au bash
exec bash

- Afficher les variables d'environnement

Avec la commande **env** ou **printenv**

- Ajouter à votre Path le chemin "/home/'votre utilisateur'/Bureau"

Qu'est-ce que la variable **PATH** sous Linux?
Dans votre fichier .**profile** ou .**login** Qu'est-ce que la variable **PATH** sous Linux ?
La variable **CHEMIN** est une variable d'environnement qui contient une liste ordonnée de chemins que Linux recherchera pour les exécutables lors de l'exécution d'une commande.
Pour la modifier.
ouvrir le fichier .bashrc dans nano et ajouter à la fin du fichier.
ici on évitera de la modifier dans le fichier .**bashrc** (problème survenu toujours inexpliqué)
PATH=$PATH::home/"votre utilisateur"/Bureau

JOB 6

L'URL de notre archive:
https://drive.google.com/file/d/1s9ZhRhjo0FXcBNRB5khAGK1jVxkZj6Uk/view?usp=sharing

On récupère l'ID du fichier
1s9ZhRhjo0FXcBNRB5khAGK1jVxkZj6Uk

wget --no-check-certificate 'https://docs.google.com/uc?export=download&id=FILE_ID' -O FILE_NAME_ALONG_WITH_SUFFIX

wget --no-check-certificate 'https://drive.google.com/uc?export=download&id=1s9ZhRhjo0FXcBNRB5khAGK1jVxkZj6Uk' -O Ghost_in_the_Shell.tar.gz

Pour un fichier ou dossier supérieur à 100Mb

wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cookies.txt --keep-session-cookies --no-check-certificate 'https://docs.google.com/uc?export=download&id=**1s9ZhRhjo0FXcBNRB5khAGK1jVxkZj6Uk**' -O- | sed -rn 's/.*confirm=([0-9A-Za-z_]+).*/\1\n/p')&id=**1s9ZhRhjo0FXcBNRB5khAGK1jVxkZj6Uk**" -O Ghost_in_the_Shell.tar.gz && rm -rf /tmp/cookies.txt

la commande suivante sert à désarchiver le document .tar nommé
tar -xzvf "archive.tar.gz "

À revoir… Malgré maintes et maintes tentatives, ces commandes n'ont pas fonctionné pour moi, j'ai cheat pour passer à la suite du projet.

JOB 7

- Créer un fichier "une_commande.txt" avec le texte suivant "Je suis votre fichier texte"
- Compter le nombre de lignes présentes dans votre fichier de source apt et les enregistrer dans un fichier nommé "nb_lignes.txt"
- Afficher le contenu du fichier source apt et l'enregistrer dans un autre fichier appelé "save_sources"
- Faites une recherche des fichiers commençant par "." tout en cherchant le mot alias qui sera utilisé depuis un fichier

Afin d'imbriquer cette suite de commande dans une seule ligne de commande, nous utiliserons les opérateurs suivant " > < >> <<| "

Le symbole **>** (supérieur à) sert à rediriger la sortie de la commande précédente vers (un fichier donné ou défini) au lieu de l'afficher sur le terminal.
Si le fichier nommé n'existe pas, il sera créé. Par contre si ce fichier existe déjà, la commande réécrira par dessus et effacera les données préexistantes.
Le symbole **>>** sert également à rediriger la sortie d'une commande précédemment exécutée mais à la différence du symbole précédent, il ne réécrit pas par dessus le fichier si ce dernier existe déjà.
Il ajoute la nouvelle sortie de la commande au contenu déjà existant dans le fichier en question.

Le symbole | (pipe) : sert à dire au Shell qu'on veut utiliser la sortie de la commande tapée à gauche, comme entrée de la commande de droite.

- `wc` counts lines, words, and characters in its inputs.
- `cat` displays the contents of its inputs.
- `sort` sorts its inputs.
- `head` displays the first 10 lines of its input.
- `tail` displays the last 10 lines of its input.
- `command > [file]` redirects a command's output to a file (overwriting any existing content).
- `command >> [file]` appends a command's output to a file.
- `[first] | [second]` is a pipeline: the output of the first command is used as the input to the second.
- The best way to use the shell is to use pipes to combine simple single-purpose programs (filters).

La commande sera la suivante:

`echo "Je suis votre fichier" > une_commande.txt | wc -l /etc/apt/sources.list > nb_lignes.txt | cat /etc/apt/sources.list > save_sources.txt | find .* -type -f -print | grep -ri "alias"`

Pour aller plus loin…

Ici nous nous servirons des opérateurs suivant : "**| || & &&**"

- Installer la commande tree
- Lancer la commande **tree** en arrière-plan qui aura pour but d'afficher toute l'arborescence de votre **/** en enregistrant le résultat dans un fichier "**tree.save**"
- Lister les éléments présents dans le dossier courant et utiliser directement le résultat de votre première commande pour compter le nombre d'éléments trouvés
- Lancer une commande pour update vos paquets, si l'update réussi alors, vous devrez lancer un upgrade de vos paquets. Si l'update échoue, votre upgrade ne se lancera pas

|| (double pipe/ OU Logique) : permet, si la 1ere commande, la seconde ne sera pas exécutée

commande1 **||** commande2

La seconde commande (**commande2**) est exécutée uniquement si le code de retour de la commande (**commande1**) est égale à **1** (faux).

Le code de retour global est égal à **0** (vrai) si au moins une des commandes retourne un code égal à **0** (vrai).

& : imbriquer une commande avec la suivante

&& (ET Logique): permet de vérifier l'état de la 1ere commande. La 2nde commande s'exécute si la 1ère est réussie
commande1 && **commande2**

La seconde commande (**commande2**) est exécutée uniquement si le code de retour de la commande (**commande1**) est égale à **0** (vrai).

Le code de retour global est égal à **0** (vrai) si le code de retour de chaque commande est égal à **0** (vrai).

sudo apt-get install tree | tree & tree -o tree.save | ls -1 | wc -l | sudo apt-get update && sudo apt-get upgrade