Shell

Job 01

Afficher le manuel de la commande ls

man ls

Afficher les fichiers cachés du home de votre utilisateur

cd ~ && ls -a | grep "^\."

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

ls -ld .*
ls -lad .*

Comment ajouter des options à une commande ?

Avec un tiret.

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

-
--

Job 02

Afficher les 10 premières lignes du fichier “.bashrc”

head -10 .bashrc

Afficher les 10 dernières lignes du fichier “.bashrc”

tail -10 .bashrc

Afficher les 20 premières lignes du fichier “.bashrc”

head -20 .bashrc

Afficher les 20 dernières lignes du fichier “.bashrc”

tail -20 .bashrc

Job 03

Installer le paquet “cmatrix”

1. $ nano /etc/apt/sources.list
2. ajouter:
    a. deb http://deb.debian.org/debian/ bullseye main
    b. deb-src http://deb.debian.org/debian/ bullseye main
3. $ sudo apt update
4. $ sudo apt install cmatrix

Lancer le paquet que vous venez d’installer

$ cmatrix

Mettre à jour son gestionnaire de paquets

$ sudo apt update

Mettre à jour ses différents logiciels

$ sudo apt upgrade

Télécharger les internets : Google

$ wget https://www.google.com/

Redémarrer votre machine

$ sudo reboot

Éteindre votre machine

$ shutdown -r now (a l’instant)
$ shutdown -r 1 (en une minute)

Job 04

Créer un fichier users.txt qui contiendra User1 et User2 séparé par un retour à la ligne

$ echo -e "User1\nUser2" > users.txt

Créer un groupe appelé “Plateformeurs”

$ sudo groupadd Plateformeurs

Créer un utilisateur appelé “User1”

$ useradd User1

Créer un utilisateur appelé “User2”

$ useradd User2

Ajouter “User2” au groupe “Plateformeurs”

$ sudo usermod -a -G Plateformeurs User2

Copier votre “users.txt” dans un fichier “droits.txt”

$ cp users.txt droits.txt

Copier votre “users.txt” dans un fichier “groupes.txt”

$ 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

$ sudo chmod -c o=r droits.txt

ou

1. $ sudo apt install acl
2. $ sudo setfacl --modify user:User2:r droits.txt
3. $ getfacl droits.txt

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

$ sudo chmod -c go=r groupes.txt

Changer les droits du fichier pour que le groupe “Plateformeurs” puissent y accéder en lecture/écriture.

1. $ sudo setfacl --modify group:Plateformeurs:6 groupes.txt
2. $ getfacl groupes.txt

Job 05

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

1. $ echo -e “alias la=’ls -la’” >> ~/.bashrc
2. $ source ~/.bashrc

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

1. $ echo -e “alias update=’apt-get update’” >> ~/.bashrc
2. $ source ~/.bashrc
3. $ sudo su
4. root $ update

Ajouter un alias qui permettra de lancer la commande “apt-get upgrade” en tapant “upgrade”

1. $ echo -e “alias upgrade=’apt-get upgrade’” >> ~/.bashrc
2. $ source ~/.bashrc
3. $ sudo su
4. root $ upgrade

Ajouter une variable d’environnement qui se nommera “USER” et qui sera égale à votre nom d’utilisateur

1. $ USER=faustina
2. $ echo $USER

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

$ source ~/.bashrc

Afficher les variables d’environnement

$ printenv

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

1. $ export PATH=”/home/faustina/Bureau:$PATH”
2. $ source ~/.bashrc
3. $ printenv $PATH

Job 06

Vous devez télécharger l’archive suivante et la désarchiver seulement avec le terminal. Cette manipulation vous permettra d’accéder à la suite du sujet.
https://drive.google.com/file/d/11dSelXQuH4tih6zesbv-6OMEpr-sT77X/view?usp=sharing

$ scp -P 2222 'Downloads/Copie de Ghost in the Shell.tar.gz' faustina@localhost:~/Downloads
$ mkdir Start/shell/Job6 && tar -xzvf 'Downloads/Copie de Ghost in the Shell.tar.gz' -C Start/shell/Job6

Ghost in the Shell

Job 07

Maintenant, vous allez approfondir les commandes, avec les caractères suivants “> < >> <<|”, votre fichier de documentation contiendra les actions ci-dessous ainsi que leur équivalent en ligne de commande Linux :

Créer un fichier “une_commande.txt” avec le texte suivant “Je suis votre fichier texte”

$ echo “Je suis votre fichier texte” >> une_commande.txt

Compter le nombre de lignes présentes dans votre fichier de source apt et les enregistrer dans un fichier nommé “nb_lignes.txt”

$ wc -l /etc/apt/sources.list | awk ‘BEGIN={FS=” ”} {print $1 > “nb_lignes.txt”}’

Afficher le contenu du fichier source apt et l’enregistrer dans un autre fichier appelé “save_sources”

$ cat /etc/apt/source.list >> save_sources

Faites une recherche des fichiers commençants par “.” tout en cherchant le mot alias qui sera utilisé depuis un fichier

$ grep -r .[^.]* *

Pour aller plus loin … Toutes les actions sont à réaliser en une seule ligne de commande. Votre fichier de documentation contiendra les actions ci-dessous ainsi que leur équivalent en ligne de commande Linux en utilisant seulement les caractères suivants “| || & &&” :

Installer la commande tree

$ sudo apt install tree

Lancer la commande tree en arrière-plan qui aura pour but d’afficher toute l'arborescence en de votre / en enregistrant le résultat dans un fichier “tree.save”

$ tree / > tree.save &

Lister les éléments présents dans le dossier courant est utilisé directement le résultat de votre première commande pour compter le nombre d’éléments trouvés

$ ls -1 | wc -l

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.

```
$ sudo apt update && yes | sudo apt upgrade
```

Shell.exe

Chacun des jobs est à créer dans un sous-dossier nommé avec le nom du job soit par exemple “Job 1”

$ for i in $(seq -w 1 9) ; do mkdir “~/Start/Job$i” ; done

Job 01

En ligne de commande : Créer un fichier nommé myfirstscript.sh, écrivez à l'intérieur votre premier script : echo “i’m a script”. Le mot clé “echo” permet d’afficher un texte et bien d’autres choses que vous verrez par la suite… Il faut maintenant donner les droits d'exécution à votre utilisateur. Une fois les droits donnés, exécutez votre script.

$ cd Job1 && file_name=myfirstscript.sh && echo ‘echo “I’\’’m a script”’ >> $file_name && sudo setfacl -m u:faustina:x $file_name && bash $file_name && cd ..

Job 02

Votre gestionnaire de paquet préféré a besoin d’être mis à jour de manière récurrente, une tâche avec 2 lignes de commandes qui pourrait être simplifié en une seule ! Réalisez un script nommé myupdate.sh qui met à jour votre gestionnaire de paquet.

En plusieurs commandes:

$ cd Job2
$ nano myupdate.sh

GNU nano 5.4 myupdate.sh
sudo apt update &&\ sudo apt upgrade -y
[[^X + Y + Entrée]]

$ bash myupdate.sh

En une ligne de commande:

$ cd Job2 && file_name=myupdate.sh && echo ‘sudo apt update’ >> $file_name && echo ‘sudo apt upgrade -y’ > $file_name && bash $file_name && cd ..

Job 03

Créer un script nommé add.sh qui prendra cette fois-ci des paramètres en entrée de script. Ce script devra permettre d’additionner 2 nombres. Les nombres doivent être renseignés en argument du script comme ceci :

$ cd Job3 && echo ‘echo $[$1 + $2]’ >> add.sh && bash add.sh 40 2 && cd ..

Job 04

Pour ce job, votre script devra créer un fichier dans le répertoire courant qui prendra le nom passé en premier argument. Il devra prendre en deuxième argument le nom d’un fichier dont il se servira pour remplir celui que vous venez de créer. Pour réaliser ce job, vous devrez utiliser obligatoirement les redirections. Votre script se nommera argument.sh et se lancera de la manière suivante : ./argument.sh myfile.txt copyfile.txt

$ cd Job4 && echo ‘cp $1 $2’ >> argument.sh && bash argument.sh myfile.txt copyfile.sh && cd ..

Job 05

Vous allez maintenant voir les conditions, pour cela il vous faut réaliser un script qui affichera soit “Bonjour, je suis un script !” soit “Au revoir et bonne journée” selon l’argument passé. Le paramètre “Hello” devra afficher le message “Bonjour .....” et “Bye” devra afficher le message “Au revoir .....”. Votre script devra se nommer hello_bye.sh et se lancera de cette façon : ./hello_bye.sh Bye.

$ nano Job5/hello_bye.sh

GNU nano 5.4 hello_bye.sh

```
#! /bin/bash
```

```
if [[ $1 == “Hello” ]] ; then
        echo “Bonjour, je suis un script”
elif [[ $1 == “Bye” ]] ; then
        echo “Au revoir et bonne journée”
fi
```

[[^X + Y + Entrée]]

$ bash Job5/hello_bye.sh Bye

Job 06

Poussons un peu les boucles, pour cela, vous allez créer une minicalculatrice qui permettra de faire les opérations suivantes : “x + - ÷”. Votre script devra se nommer my_calculator.sh. Les chiffres de l’opération seront passés en premier et troisième paramètre et le symbole de l’opération en deuxième position de telle sorte que votre script se lance de la manière suivante : ./my_calculator.sh 2 + 3

$ nano Job6/my_calculator.sh

GNU nano 5.4 my_calculator.sh

```
#! /bin/bash

awk “begin{print $*}”;
```

[[^X + Y + Entrée]]

$ bash Job6/my_calculator.sh 2 + 3

Job 07

Il est désormais temps d’entrevoir le monde des boucles. Pour cela, vous allez créer un script nommé myloop.sh qui va afficher 10 fois la phrase suivante et afficher en fin de phrase un chiffre qui s’incrémente à chaque fois :
“Je suis un script qui arrive à faire une boucle 1”
“Je suis un script qui arrive à faire une boucle 2”
“Je suis un script qui arrive à faire une boucle 3”
“Je suis un script qui arrive à faire une boucle 4”
.....
“Je suis un script qui arrive à faire une boucle 10”

Vous devez obligatoirement utiliser les boucles pour réaliser ce script.
$ nano Job7/myloop.sh

GNU nano 5.4 myloop.sh

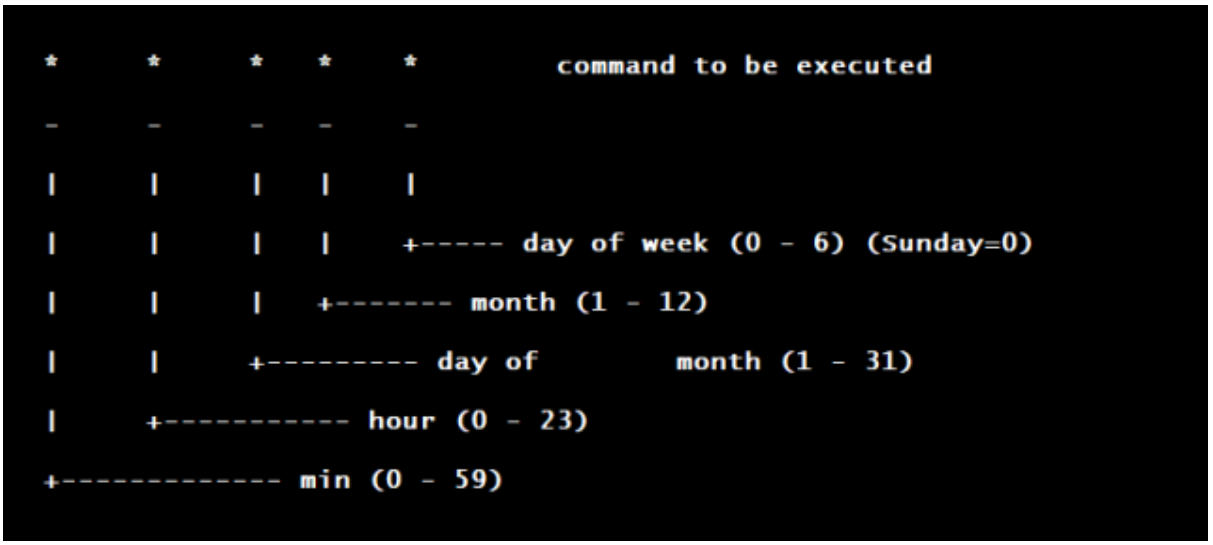
```
#! /bin/bash

for i in {1..10} ;
do
        echo “Je suis un script qui arrive à faire une boucle $i”
done
```

[[^X + Y + Entrée]]

$ bash Job7/myloop.sh

Job 08

Maintenant que vous connaissez les boucles, vous devez utiliser les **Cron** pour permettre d’exécuter ce script toutes les heures.



Ce script aura pour but d’extraire de vos logs Linux le nombre de connexions à votre session qui ont eu lieu sur votre ordinateur. Ce nombre sera écrit dans un fichier qui se nommera **number_connection-Date** où **Date** sera remplacé par la date de création de votre fichier avec l’heure sous le format **jj-mm-aaaa-HH:MM**.

$ cd Job8 && number_connections=$(last | grep -c $USER) && current_date=$(date +’%d-%m-%Y-%H:%M’)

```
$ filename=”number-connection-$current_date”
$ echo $number_connections >> $filename
```

Par la suite, ce fichier devra être archivé avec tar et déplacé dans un sous-dossier appelé **Backup**.

```
$ backup_folder=Backup && mkdir -p “$backup_folder”
$ tar -czf “$backup_folder/$filename.tar.gz” “$filename”
```

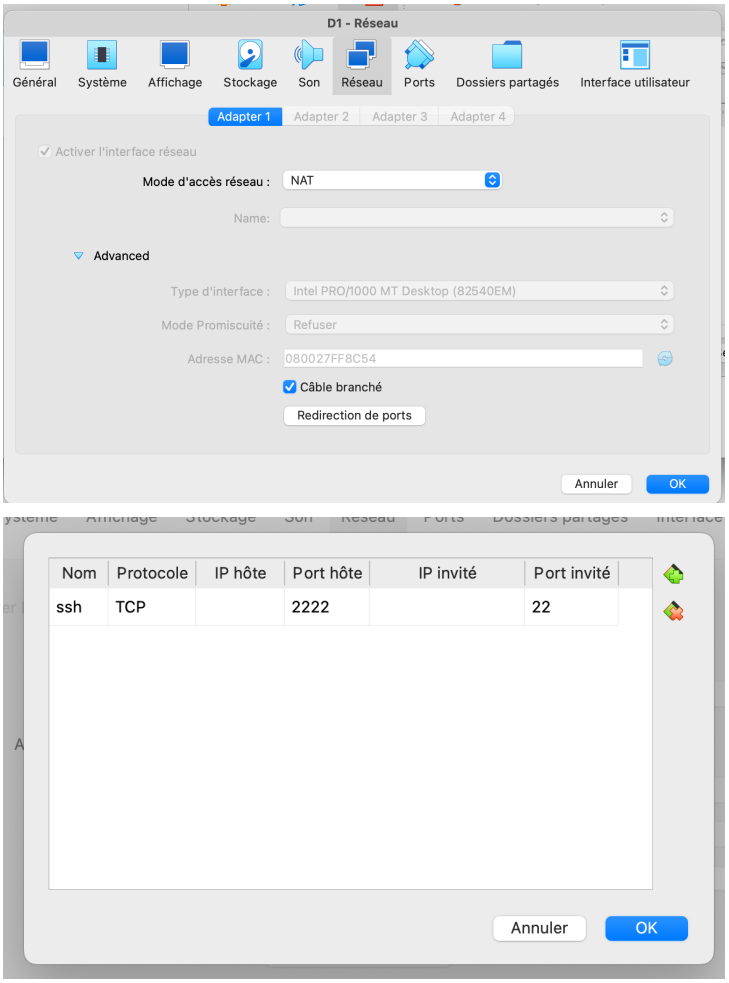
Votre fichier script devra se nommer **get_logs.sh**. Votre arborescence sera donc *Job8* → *Backup* → ***number_connection-Date***

N’oubliez pas de le lancer via les **Cron** de votre ordinateur.

```
$ crontab -e
0 * * * * bash ~/Start/shell.exe/get_log.sh
```

Job 09

Pour accéder à la machine virtuelle, il faut configurer



Se connecter à la machine virtuelle depuis la machine hôte:

$ ssh -p 2222 faustina@localhost

Copier un fichier dans la machine virtuelle depuis la machine hôte:

$ scp -P 2222 fichier_a_copier.extension faustina@localhost:~/

Créer un script nommé **accessrights.sh** qui depuis ce fichier CSV, récupère les informations des utilisateurs et les crée sur votre système.

$ scp -P 2222 Downloads/Shell_Userlist.csv faustina@localhost:~/
$ awk -F, ' FNR>=2 { print "ID:", $1, "Utilisateur:", $2, $3, "MDP:", $4, "Role:", $5 }' Shell_Userlist.csv (pour lire le fichier)

Si l’utilisateur est un **admin.**, donnez-lui le rôle de **super utilisateur (root)** de votre système.

$ nano accessrights.sh

GNU nano 5.4 accessrights.sh

```
#! /bin/bash
csv_file="/home/faustina/Start/shell.exe/Job9/Shell_Userlist.csv"

remove_spaces() {
    local str="$1"
    echo "${str}" | awk '{$1=$1};1'
}

create_update_user() {
    read
    while IFS=, read -r id prenom nom passwd role || [ -n "$id" ]; do
        name=$(remove_spaces "$prenom")_$(remove_spaces "$nom")
        role=$(remove_spaces "$role")

        if [ "$role" = "Admin" ]; then
            group="root"
        else
            group="users"
        fi

            if id "$name" >/dev/null 2>&1 ; then
        sudo usermod "$name" -u "$id" -p "$4" -aG "$group"
            else
            sudo useradd "$name" -u "$id" -p "$4" -g "$group"
            fi
    done <"$csv_file"
}

check_changes() {
    current_timestamp=$(stat -c %Y "$csv_file")
    previous_timestamp=$(cat "/var/tmp/previous_csv_timestamp" 2>/dev/null || echo 0)

    if [ "$previous_timestamp" = "$current_timestamp" ]; then
        exit 0
    else
            echo "$current_timestamp" > "/var/tmp/previous_csv_timestamp"
            create_update_user
    fi
}

check_changes >> /var/log/accessrights.log 2>&1
```

[[^X + Y + Entrée]]

Pour la suite, utilisez les cron pour permettre au script de se relancer automatiquement s'il y a un changement dans le fichier CSV. (Pour tester, je vous invite à modifier le fichier à la main).

$ crontab -e
* * * * * /bin/bash -c "/home/faustina/Start/shell.exe/Job9/accessrights.sh"