

Job 1 :

Comment ajouter des options à une commande ?

On ajoute des options par le biais d'un "-" comme pour : -a , -l ou encore -d.

Ces options ont pour but de modifier les effets d'une ou plusieurs commandes sur le terminal.

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

Les deux syntaxes principales d'écriture sont :

-Bash

-Ksh

Ce sont tous deux des langages de programmation de commandes Linux ou Unix. Cependant Ksh est un peu plus ancien et a quelques limites donc moins d'utilisateurs tandis que le Bash est bien plus récent, il agit comme une extension du KSH car il incorpore les mêmes fonctionnalités mais avec quelques options supplémentaires.

Commandes pour voir les fichiers/ dossiers cachés :

"ls" Pour lister dans un terminal les éléments non-cachés du dossier en cours,

Pour afficher tous les éléments, y compris les éléments cachés, il suffit d'ajouter l'argument -a («all» en anglais):

"ls -a" pour voir tous les dossiers avec leurs fichiers à l'intérieur

Et pour n'afficher que les fichiers et dossiers cachés:

"ls -d .*"

Si vous ajoutez /, vous ne voyez que les dossiers cachés:

"ls -d .*/"

Pour voir les droits d'un fichier :

"ls -la"

Job 2 :

Pour afficher les premières lignes d'un fichier :

```
simon@simon-Ubuntu:~$ sudo head -n 20 .bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
    *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000
```

`sudo head [-n nombre de lignes] [nom du fichier]`

par exemple : `sudo head -n 20 .bashrc` : cela affichera les 20 premières lignes du fichier.

Pour afficher les premières lignes d'un fichier :

`sudo tail [-n nombre de lignes] [nom du fichier]`

par exemple : `sudo tail -n 20 .bashrc` : cela affichera les 20 dernières lignes du fichier

Job 3 :

- Installer le paquet "cmatrix" : `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 full-upgrade`
- Télécharger les internets : Google

1. `wget`

[https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.
deb](https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb)

2. `sudo apt install ./google-chrome-stable_current_amd64.deb`

- Redémarrer votre machine : `reboot`
- éteindre votre machine : `shutdown -h now`

Job 4 :

créer un fichier : `echo "contenu">nom_fichier.txt`

copier le contenu d'un fichier vers un autre : `cat "A" > "B"`

- Créer un groupe appelé "Plateformeurs" : `sudo groupadd Plateformeurs`
- Créer un utilisateur appelé "User1" : `sudo useradd user 1`
- Créer un utilisateur appelé "User2" `sudo useradd user 2`
- Ajouter "User2" au groupe Plateformeurs : `usermod -g plateformeurs user2`
- Copier votre "users.txt" dans un fichier "droits.txt" : `cat "users.txt" > "droits.txt"`
- Copier votre "users.txt" dans un fichier "groupes.txt" `cat "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 -R 740 droits.txt**
- Changer les droits du fichier "groupes.txt" pour que les utilisateurs puissent accéder au fichier en lecture uniquement : **sudo chmod -R 754 groupes.txt**
- Changer les droits du fichier pour que le groupe "Plateformeurs" puissent y accéder en lecture/écriture. : **sudo chmod -R 760 groupes.txt**

Job 5 :

- 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="sudo apt-get update"

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

alias update="sudo apt-get upgrade"

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

export Simon=USER (pour créer la variable)

echo \$Simon (pour vérifier que la variable a été créée)

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

update

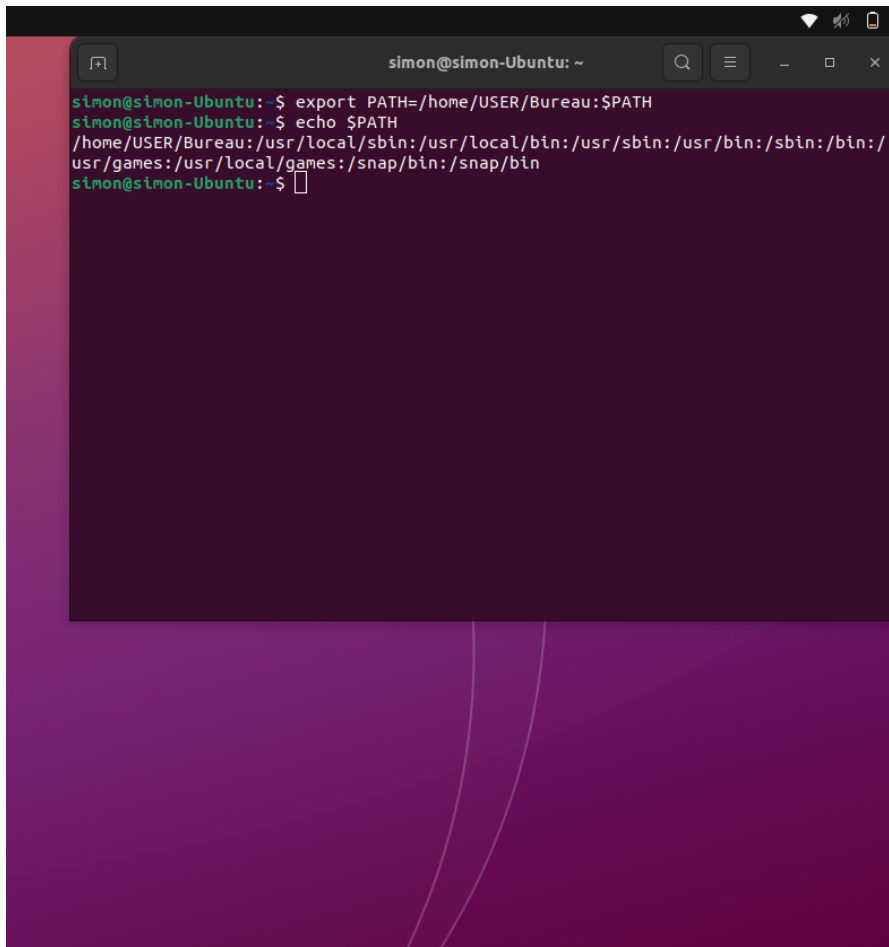
- Afficher les variables d'environnement

env

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

export PATH=/home/USER/Bureau:\$PATH (pour ajouter un chemin)

echo \$PATH (pour vérifier que le chemin a été créé).

A terminal window titled 'simon@simon-Ubuntu: ~' with search, menu, and window control icons. The terminal shows the following commands and output:

```
simon@simon-Ubuntu: ~$ export PATH=/home/USER/Bureau:$PATH
simon@simon-Ubuntu: ~$ echo $PATH
/home/USER/Bureau:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin
simon@simon-Ubuntu: ~$
```

Job 6 :

Allez dans le répertoire avec le fichier puis tapez : `tar -xzf Ghost/ in/ the/ Shell.tar.gz`

Job 7 :

`cd sources.list`

(`wc -l "nom du fichier"` pour savoir le nombre de ligne)

-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" `il y a 50 lignes (wc -l "nom du fichier" pour savoir le nombre de ligne)`

-Afficher le contenu du fichier source apt et l'enregistrer dans un autre fichier

appelé :`"save_sources" cat "sources.list" pour afficher puis cat "sources.list">"save_sources"`

-Faites une recherche des fichiers commençant par "." tout en cherchant le mot

alias qui sera utilisé depuis un fichier : `find "."`

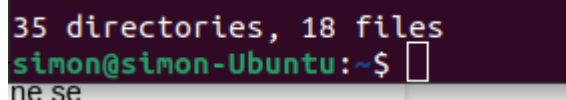
`w`

Aller plus loin...

-Installer la commande tree `sudo apt-get 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" : `sudo nohup tree & (pour lancer la commande en arrière plan) puis cp nohup.out tree.save (pour copier l'arborescence de tree)`

lister les éléments présents dans le dossier courant est utilisé directement le résultat de votre première commande pour compter le nombre d'éléments trouvés **35 répertoires et 18 fichiers**



```
35 directories, 18 files
simon@simon-Ubuntu:~$
```

-Lancer une commande pour update vos paquets, si l'update réussit alors, vous devrez lancer un upgrade de vos paquets. Si l'update échoue, votre upgrade ne se lancera pas : `sudo apt update && sudo apt full-upgrade -y`

Questions à répondre dans la documentation à la suite dans une section BONUS:

Quel est l'intérêt d'utiliser SSH ?

-Est-ce que les clés générées par SSH par défaut sont assez sécurisées ? Justifier votre réponse
Les clés générées par défaut par SSH ne sont pas assez sécurisées.

En effet, pour maximiser la sécurité il est nécessaire de changer le port d'écoute (22 par défaut) qui est connu de tous les hackers.

De plus, il est très important de désactiver l'utilisateur root qui possède tous les droits sur le système donc en cas d'intrusion dans le système notre SSH sera par conséquent compromis.

Enfin, pour avoir une sécurité optimale il est préférable de spécifier les ip des utilisateurs pouvant se connecter en SSH avec l'option AllowUsers.

-Citez d'autres protocoles de transfert ? Quelles sont les différences entre ses protocoles ?

Il y a plusieurs protocoles de transferts avec différents niveaux de sécurité.

Il y a l'AS (2,3,4), ce sont des protocoles qui servent à transférer des données confidentielles relativement sensibles de façon sécuritaire en utilisant des certificats numériques et des normes de chiffrement.

Le SCP qui est un ancien protocole utilisant le transfert de données entre hôtes de réseaux, il prend également en charge les fonctions de chiffrement et d'authentification.

Le HTTP utilise un protocole secondaire appelé TCP, il agit de façon à ce que chaque commande soit exécutée indépendamment et qu'aucune information de session ne soit conservée par le destinataire.

Le HTTPS fonctionne de la même façon que le HTTP sauf qu'il intègre des fonctions de chiffrement par TLS ou SSL.