

Shell

Job 1 : La syntaxe d'une ligne de commande bash

- Comment ajouter des options à une commande ?

Lorsque l'on insère une commande telle que "`ls`", il est possible de rajouter des options avec la syntaxe suivante :

```
commande -option1 -option2
```

À noter que les majuscules et minuscules peuvent avoir des effets différents selon la commande.

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

Les options peuvent être écrites comme ceci :

```
-optioncourte  
-optionlongue
```

Chaque option courte a un équivalent en option longue.

Job 2 : Lire “.bashrc”

Le shell permet de lire un fichier nommé “.bashrc” qui est un script shell exécuté chaque fois qu’un utilisateur ouvre un nouveau shell. Nous allons voir différentes commandes qui permettent d’imprimer le contenu de ce fichier dans le terminal.

```
cat ~/.bashrc
```

La commande “**cat**” imprime simplement le contenu du fichier dans le terminal tel qu’il est. Son principal défaut cependant est qu’il prend de la place entre les lignes de commande précédentes et suivantes. La commande suivante permet de contourner ce problème.

```
less ~/.bashrc
```

La commande “**less**” remplit la même fonction que “**cat**” à ceci près que “**less**” masque les autres lignes de commandes temporairement le temps que l’utilisateur lise le contenu du fichier ouvert. Il permet ainsi de “nettoyer” le terminal une fois la lecture terminée car, dès que l’on presse la touche “Q” (pour “quit”), le contenu du fichier disparaît du terminal qui ré-affiche alors les précédentes lignes de commande dans l’ordre.

```
head ~/.bashrc
```

La commande “**head**” permet de ne lire que les dix premières lignes d’un fichier.

```
tail ~/.bashrc
```

La commande “**tail**” permet de ne lire que les dix dernières lignes d’un fichier.

```
head ~/.bashrc -n20
```

L’option “**-nX**” (remplacer “**X**” par un chiffre) rajoutée à la fin d’une ligne de commande “**head**” permet de lire le nombre souhaité de lignes au lieu des 10 lignes habituelles.

```
tail ~/.bashrc -n20
```

L’option “**-nX**” (remplacer “**X**” par un chiffre) rajoutée à la fin d’une ligne de commande “**tail**” permet de lire le nombre souhaité de lignes au lieu des 10 lignes habituelles.

Job 3 : Installation d'un paquet et mises à jour

À présent, nous allons apprendre à installer un paquet.

```
sudo apt install cmatrix
```

Cette commande permet d'installer le paquet "cmatrix" contenant le programme du même nom.

```
cmatrix
```

Nous pouvons à présent tester le programme en question. Appuyer sur "Q" ramène à l'écran précédent.

```
sudo apt update && sudo apt upgrade
```

Il est important de mettre à jour ses paquets régulièrement, pour ce faire nous allons utiliser cette double commande. Elle permet de mettre à jour à la fois la liste des paquets pour savoir quelles sont les nouvelles versions sorties ("update") pour que le gestionnaire sache quelles mises à jour sont disponibles, et de mettre à jour tous les paquets concernés automatiquement ("upgrade").

```
sudo apt install snapd  
sudo snap install firefox  
firefox
```

Il est possible de télécharger le navigateur Firefox directement depuis le shell, mais pour cela, il faut d'abord installer "snapd" (ce qui signifie "Snap Daemon"). Il s'agit d'un gestionnaire de paquets nommés "snaps". Une fois l'installation de snapd terminée, on peut exécuter la deuxième commande qui permet d'installer Firefox, un des navigateurs les plus connus du monde. Une fois l'installation terminée, il suffit de taper "firefox" dans le terminal pour lancer l'application et la tester. Ma recherche "test" préférée est "Star Wars" :)

```
shutdown  
shutdown -r now
```

La commande "shutdown" permet d'arrêter le système après un minuteur de 60 secondes. Nous pouvons non seulement abrégé ce processus avec "now", mais également demander à Debian de redémarrer le système avec "-r".

Job 4 : Gestion des permissions

```
touch users.txt
```

La commande “**touch**” permet de créer des fichiers vides avec le nom que l’on souhaite.

```
sudo groupadd Plateformeurs
```

La commande “**groupadd**” permet de créer un groupe d’utilisateurs ce qui permet de gérer les permissions des utilisateurs en question sans avoir à les éditer un par un. On peut utiliser n’importe quel nom de groupe qui n’est pas déjà pris.

```
sudo adduser User1
```

```
sudo adduser User2
```

À présent, nous pouvons également créer les utilisateurs. Nous allons les nommer “User1” et “User2”.

```
sudo usermod -a -G Plateformeurs User 2
```

Pour rajouter User2 au groupe, la commande “**usermod**” avec les options “**-a**” (pour ajouter un membre à un groupe) et “**-G**” (pour définir le ou les groupes dans lesquels le membre sera rajouté) est employée.

```
cp users.txt droits.txt
```

```
cp users.txt groupes.txt
```

La commande “**cp**” permet simplement de copier un fichier déjà existant vers un nouveau fichier que l’on peut directement nommer. Ici, “users.txt” est copié deux fois en tant que “droits.txt” et “groupes.txt”.

```
sudo chown User1 droits.txt
```

La commande “**chown**” permet de transférer l’appartenance d’un fichier à un autre utilisateur, dans le cas présent nous avons défini que le fichier “droits.txt” appartenait à présent à “User1”. Elle ne peut être exécutée qu’avec **sudo** dû à la nature potentiellement dangereuse d’une telle manipulation.

```
sudo setfacl -m u:user2:r droits.txt
```

```
sudo setfacl -m o:r groupes.txt
```

```
sudo setfacl -m g:Plateformeurs:rw groupes.txt
```

La commande `setfacl` a comme fonction de changer les permissions d'écriture et/ou de lecture ainsi que d'exécution d'un fichier vis-à-vis d'utilisateurs spécifiques, de groupes, ainsi que de tous les autres utilisateurs, dans cet ordre de "hiérarchie" ("user">"group">"others").

L'option `-m` est nécessaire pour modifier les permissions d'un fichier.

Pour commencer à entrer les "arguments" (c'est-à-dire les paramètres de notre commande), nous avons trois possibilités de "cibles" :

`u:[username]` permet de viser un utilisateur spécifique

`g:[group]` permet de viser un groupe spécifique

`o` permet de viser tous les autres utilisateurs

Il s'agit à présent de définir les permissions pour chacune de ces cibles directement après les avoir choisies.

`:r` signifie "read-only" ou "lecture seule", le contenu du fichier pourra être consulté, mais impossible de le modifier ou de l'exécuter s'il s'agit d'un fichier exécutable.

`:rw` signifie "read and write", car en effet, il est très difficile d'écrire dans un fichier si on ne peut pas le lire en premier lieu :)

`:rwx` signifie "read, write, and execute", ce qui, comme vous l'avez probablement compris, donne toutes les permissions.

`:rx` implique que l'on peut exécuter et lire le contenu du fichier, mais pas le modifier, pratique pour empêcher des utilisateurs d'éditer un fichier et de le casser par accident

Il est donc en effet possible de combiner les lettres "r", "w" et "x" pour obtenir des permissions différentes, mais toujours dans le même ordre cependant: "r">"w">"x"

Job 5 : Créer des alias et des variables d'environnement

Nous allons à présent apprendre à créer des alias, qui sont des versions alternatives de commandes qui permettent de raccourcir et fluidifier ces dernières.

```
sudo apt install nano
```

Premièrement, il nous faut installer l'éditeur de texte "Nano" qui tourne directement dans le terminal de notre choix. D'autres éditeurs de textes tels que Vim fonctionnent également. À présent, nous devons rouvrir ".bashrc", mais cette fois-ci avec l'éditeur que nous venons d'installer :

```
nano ~/.bashrc
```

Il faut à présent aller à la toute fin du fichier dans l'éditeur de texte en s'aidant des commandes décrites en bas, puis écrire les lignes suivantes :

```
alias la="ls -la"
```

```
alias update="sudo apt-get update"
```

```
alias upgrade="sudo apt-get upgrade"
```

Comme vous l'avez probablement compris, "alias" est le point de départ pour créer un alias. Nous rentrons après l'alias en question, par exemple "upgrade", suivi de "sudo apt-get upgrade" pour que l'alias "upgrade" exécute directement la commande "sudo apt-get upgrade".

Nous pouvons à présent fermer l'éditeur de texte sans oublier de sauvegarder.

```
USER=[username]
```

Cette commande a pour but de définir la variable d'environnement USER en tant que votre nom d'utilisateur. À l'origine, la variable USER est égale à "root". Bien évidemment, pensez à remplacer "[username]" par votre véritable nom d'utilisateur.

```
source ~/.bashrc
```

Nous allons à présent employer la commande "source" pour que le terminal actualise ".bashrc" et rende l'utilisation des nouveaux alias possible.

```
PATH=$PATH:/usr/local/bin:/usr/bin:/usr/local/games:/usr/games:/snap/bin:/home/[username]/Bureau
```

```
export PATH
```

Nous allons à présent rajouter le chemin (ou “path”) du bureau dans la variable PATH
Utiliser “`printenv`” pour voir quel est la valeur actuelle de la variable est très pratique car cela permet non seulement de copier-coller la valeur actuelle et d’y rajouter “`/snap/bin:/home/[username]/Bureau`”, mais également de retrouver la valeur originale dans le cas où l’utilisateur fait une fausse manipulation qui pourrait avoir des conséquences assez graves, parmi lesquelles le fait que la plupart des commandes cessent de fonctionner.

```
printenv
```

Nous pouvons à présent utiliser “`printenv`” pour vérifier que tout a été correctement enregistré.

Job 6 : Extraction d'une archive tar

```
tar -xf 'Ghost in the Shell.tar'
```

L'extraction d'une archive tar est très simple. Une fois téléchargée, il faut exécuter la commande `tar` suivie de l'option `-xf` qui indique qu'il faut extraire une archive, que l'on doit ensuite nommer. Ici, il s'agit de `'Ghost in the Shell.tar'`. Nous pouvons également utiliser l'option `-xvf` qui imprime le nom des différents fichiers qui viennent d'être extraits.

```
sudo apt-get install mupdf
```

```
mupdf 'Ghost in the Shell.pdf'
```

L'extraction révèle des fichiers PDF. Pour les lire, nous allons installer mupdf, qui est un lecteur de PDF minimaliste, mais suffisant pour les besoins du projet.

Job 7 : Commandes multiples

(note : je ne suis pas parvenu à compléter ce job)

```
echo Je suis un fichier texte > une_commande.txt
```

Le symbole ">" signifie qu'il faut sauvegarder le résultat affiché par le terminal dans un fichier, en l'occurrence un fichier texte.

```
wc -l < /etc/apt/sources.list > nb_lignes.txt
```

wc permet de compter les lignes, -l permet de retourner un résultat sur le terminal, < indique quel fichier est concerné.

```
ls -ld .*
```

Pour aller plus loin

(note : je ne suis pas parvenu à compléter ce job)

```
sudo apt-get install tree && tree / > tree.save
```

`&&` permet de faire exécuter la deuxième commande si la première réussit.

```
ls
```

```
sudo apt-get update && sudo apt-get upgrade
```