TP d'initiation au Shell

20 mars 2016

1 Manipulation de répertoires de fichiers

Lorsque vous ouvrez une session, votre répertoire de travail est votre dossier personnel. Vous avez vu en cours que la commande `pwd` (*Print Working Directory*) vous permet d'afficher le chemin absolu du répertoire actuel.

Q. 1: Vérifiez que votre répertoire de travail est bien votre `home`.

Solution 1: Commande `pwd`.

Q. 2: La commande `ls` permet de lister le contenu d'un répertoire. Les options suivantes sont couramment utilisées :

`-a` : affiche les dossiers cachés

`-l` : affiche les détails (type, droits, propriétaire, taille etc...)

Il est possible de grouper les options. Ainsi, la commande `ls -a -l` peut être écrite `ls -al`. Listez le contenu de votre `home`, avec puis sans les détails.

Solution 2: `ls ~ && ls -al ~`.

Q. 3: Le symbole ~ signifie *home directory of*. Listez le contenu du `home` de votre voisin, en utilisant `~IDduvoisin`.

Solution 3: `ls ~IDduvoisin`

Q. 4: Affichez la liste des fichiers contenus dans le dépôt du groupe 1A, sans vous déplacer dans le répertoire. Proposez deux solutions, l'une avec un chemin absolu et l'autre avec un chemin relatif.

Solution 4:

Q. 5: Dans votre dossier `home`, créez un répertoire nommé `TP_UNIX` à l'aide de la commande `mkdir` (*Make Directory*).

Solution 5: `cd ~ && mkdir TP_UNIX`

Q. 6: La commande `cd` (*Change Directory*) permet de se déplacer dans l'arborescence. Elle prend en paramètre un chemin, absolu ou relatif. Ce chemin peut être auto-complété à l'aide de la touche `TAB`. Entrez dans le répertoire créé à la question précédente.

Solution 6: `cd TP_UNIX`

2 Manipulation du contenu d'un fichier

Q. 7: Pour créer un fichier (vide), on peut utiliser la commande `touch`. Créez un fichier nommé *awesomefile.txt*.

Solution 7: `touch` awesomefile.txt

Q. 8: La commande `echo` permet d'écrire sur la sortie standard. Essayez-la avec le paramètre `coucou`.

Solution 8: `echo coucou`

Q. 9: On peut également rediriger la sortie de `echo` vers un fichier, ce qui permet d'écrire dans celui-ci. On utilise pour cela un chevron >. Cela écrase le contenu du fichier ; si l'on désire simplement ajouter du contenu à la fin du fichier, on utilise deux chevrons, >>. Écrivez *coucou* dans *awesomefile.txt*.

Solution 9: `echo coucou >` awesomefile.txt

Q. 10: Vérifier le résultat de la question précédente en utilisant la commande `cat`.

Solution 10: `cat` awesomefile.txt

Q. 11: `echo` n'est pas la seule commande dont la sortie peut être redirigée. Écrivez le manuel de la commande `cat`, dans *awesomefile.txt*.

Solution 11: `man cat >` awesomefile.txt

Q. 12: Pour des fichiers dépassant une certaine taille, la commande `cat` est peu pratique. Vérifiez le résultat de la question précédente à l'aide de `less`.

Solution 12: `less` awesomefile.txt

Il existe également des éditeurs de textes accessibles en console ; on peut citer *nano*, *vim*, ou encore *emacs*.

3 Manipulation de fichiers

Q. 13: À l'aide de la commande `cp` (*copy*), copiez *awesomefile.txt* dans votre `home`. Vérifiez le résultat avec `ls`.

Solution 13: `cp` awesomefile.txt `../copied.txt && ls ..`

Q. 14: La commande `mv` (*move*) permet de déplacer/renommer des fichiers. Renommez le fichier que vous venez de copier dans votre `home`, sans oublier de consulter le manuel.

Solution 14: `mv ../`awesomefile.txt `../newName.txt`

Q. 15: Supprimez maintenant ce fichier, à l'aide de `rm` (*remove*).

Solution 15: `rm ../newName.txt`

Q. 16: Qu'aurait-il fallu changer dans les questions précédentes si l'on avait manipulé un dossier (vide/non vide) ?

Solution 16:

`cp` : rajouter `-r`

`mv` : rien

`rm` : rajouter `-r`, ou utiliser `rmdir` si vide

4 Gestion des droits d'accès

Pour gérer l'aspect multi-utilisateurs, les fichiers/dossiers Unix disposent de droits d'accès :

r Lecture

w Écriture

x Exécution

Les trois peuvent être différents suivant 3 niveaux :

u L'utilisateur

g le groupe, sauf l'utilisateur

o tout le monde, sauf le groupe et l'utilisateur

Pour un dossier, le droit d'exécution correspond au droit d'accès, tandis que les droits de lecture et d'écriture correspondent l'accès à la liste des fichiers s'y trouvant et à la création de nouveaux fichiers.

Pour visualiser ces droits, on utilise la commande `ls -l`. On obtient alors un résultat de la forme de la figure 1. On s'intéresse alors à la première colonne, à lire suivant les informations de la figure 2.

Pour changer les droits d'un fichier/dossier, on utilise la commande `chmod`.

Q. 17: Créez un fichier nommé *test.sh*, et écrivez-y la commande `echo coucou` (par la méthode de votre choix). *test.sh* est alors un script shell, potentiellement exécutable, qui écrira "coucou" à chaque fois qu'il sera lancé (par la commande `./test.sh`). Notez les droits actuels du fichier. Avez-vous ceux nécessaires pour l'exécuter ?

Solution 17: `echo "echo coucou" > test.sh ls -l -rw-r--r--` -¿ non.

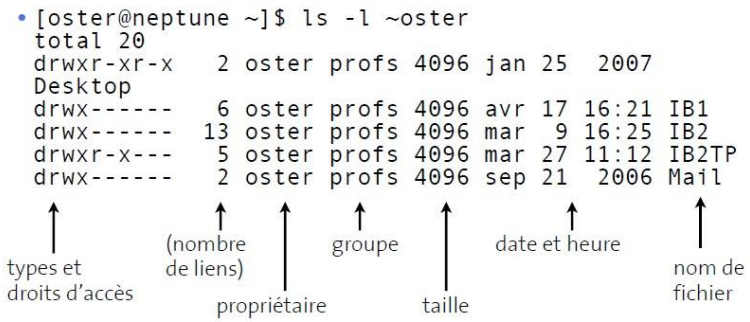

FIGURE 1 – Résultat de `ls -l`

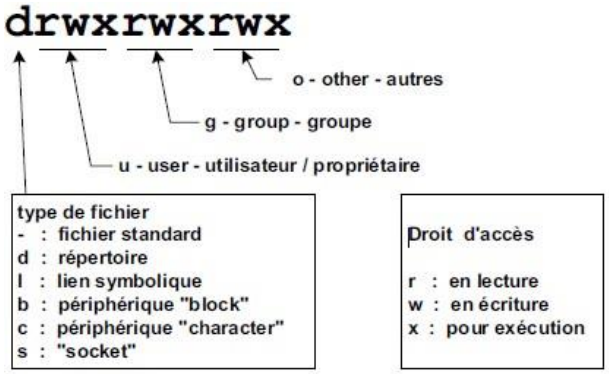

FIGURE 2 – Légende des droits d'accès

Q. 18: Il existe plusieurs manières d'utiliser `chmod`. Entrez la commande suivante : `chmod u+x test.sh`. Que remarquez-vous ? Consultez le manuel et expliquez `u+x`.

Solution 18: L'utilisateur peut maintenant exécuter `test.sh`.

Q. 19: Vous avez sans doute remarqué dans le manuel la deuxième manière d'utiliser `chmod`. Il s'agit de l'écriture *octale* des droits d'accès. Entrez la commande `chmod 777 test.sh`, puis visualisez les nouveaux droits de `test.sh`. Que remarquez-vous ?

Solution 19: Tout le monde a tous les droits.

5 Gestion des processus

Q. 20: Que fait la commande `ps` ? Essayez-là.

Solution 20: `ps` liste les processus en cours.

Lorsque l'on lance une commande, le shell ne redonne pas la main avant que celle-ci ne se soit terminée. On peut cependant terminer ou arrêter momentanément l'exécution d'une commande, à l'aide des raccourcis suivant : **Ctrl-c** (fin) et **Ctrl-z** (pause). Le shell rend alors la main.

Si le processus a été mis en pause, il est alors possible de relancer son exécution, en premier plan (on reperd alors la main) ou en arrière plan (on garde alors la main). On utilise pour cela les commandes `bg` (*background*) et `fg` (*foreground*).

Il est également possible de lancer une commande directement en arrière plan, en la terminant avec le signe `&`.

On peut utiliser la commande `jobs` pour savoir quels processus sont actuellement en arrière plan.