TP 8 - Révision générale

Objectif : vous préparer à l'examen de vendredi. Les questions sont nouvelles, mais font appel aux mêmes connaissances. L'examen sera *open book* : entraînez-vous à retrouver une information dans vos notes et avec `man` ou `--help`. Ce TP n'est pas noté et il n'y a rien à rendre.

Organisation conseillée : périodes 1 à 3, travail au terminal, avec vérification de vos résultats ; période 4, mini-examen individuel sur papier. Si une question bloque plus de quelques minutes, passez à la suivante et revenez-y ensuite.

Période 1 – Retrouver les réflexes

Exercice 1 (Commandes et système)

(a) Associez chaque commande à son rôle : `pwd`, `mkdir`, `ps`, `kill`, `grep`, `wc`, `head`, `chmod`, `apt`, `du`. Les rôles sont : afficher le répertoire courant ; créer un répertoire ; afficher les processus ; envoyer un signal ; rechercher des lignes ; compter des lignes ou des mots ; afficher les premières lignes ; modifier des permissions ; gérer les paquets ; mesurer l'espace disque utilisé.

(b) Expliquez en une phrase : la différence entre un chemin absolu et relatif ; le rôle de `root` ; ce qu'est un processus ; le rôle de `PATH` ; le rôle d'un gestionnaire de paquets.

(c) Sans modifier le système, trouvez une commande permettant de consulter les processus et une commande donnant la taille occupée par votre répertoire personnel. Consultez l'aide si nécessaire.

Exercice 2 (Chemins, copies et archives)

Dans un terminal, préparez le terrain suivant :

```
mkdir -p ~/revision08/documents/{cours,exercices}
mkdir -p ~/revision08/images
cd ~/revision08/documents/cours
printf 'Introduction Linux\n' > notes.txt
```

(a) Rejoignez `exercices` avec un chemin relatif, puis `images`, toujours avec un chemin relatif.

(b) Revenez dans `cours`. Copiez `notes.txt` dans `exercices` sous le nom `notes_copie.txt`, puis renommez cette copie en `ancien_cours.txt`.

(c) Depuis `~/revision08`, créez une archive compressée `documents.tar.gz` du répertoire `documents`, puis extrayez-la dans un répertoire `restauration` créé pour l'occasion. Vérifiez le résultat.

Période 2 – Permissions et flux

Exercice 3 (Lire et modifier des permissions)

Considérez la ligne suivante :

```
-rw-r----- 1 clara equipe 820 Nov 4 09:00 rapport.txt
```

(a) Donnez le propriétaire, le groupe propriétaire et les droits respectifs du propriétaire, du groupe et des autres.

(b) Écrivez une commande qui donne au propriétaire tous les droits, au groupe lecture et exécution, et aucun droit aux autres. Donnez les permissions finales affichées par `ls -l`.

(c) Un fichier ordinaire possède les droits `-rwxrwxrwx`. Écrivez une commande pour obtenir `-rw-r-----`. Expliquez le nombre octal utilisé.

Exercice 4 (Redirections et erreurs)

(a) Quel est le contenu final de `journal.txt` après ces commandes ?

```
echo "un" > journal.txt
echo "deux" >> journal.txt
echo "trois" >> journal.txt
echo "quatre" > journal.txt
```

(b) Expliquez ce qui se retrouve dans `ok.txt` et `erreurs.txt` :

```
ls /tmp /chemin_qui_n_existe_pas > ok.txt 2> erreurs.txt
```

(c) Écrivez une commande qui trie le contenu de `noms.txt` en utilisant explicitement `<` pour l'entrée et `>` pour enregistrer la sortie dans `ordre.txt`.

Exercice 5 (Construire et réparer des pipelines)

Créez le fichier `ventes.txt` :

```
Alice:12
Bob:07
Claire:12
David:19
Emma:07
Farid:04
```

(a) Avec `cut`, `sort`, `uniq` et `head`, affichez les deux plus petites valeurs distinctes, en ordre croissant, une par ligne.

(b) Un étudiant écrit `grep WARNING journal.log > wc -l` pour compter les lignes contenant `WARNING`. Expliquez l'erreur, puis corrigez avec un pipe.

Période 3 – Regex, sed et Bash

Exercice 6 (Expressions régulières)

(a) Expliquez les symboles `^`, `.*` et `$` dans une expression régulière.

(b) Un fichier `prenoms.txt` contient : `ana`, `adam`, `alina`, `bob`, `amelia`, chacun sur une ligne. Qu'affiche `grep '^a.*a$' prenoms.txt` ?

(c) Écrivez une commande `grep -E` qui sélectionne dans `references.txt` les lignes composées d'exactement trois lettres majuscules suivies de deux chiffres.

(d) Le fichier `villes.txt` contient des lignes comme `Alice;Bruxelles`. Écrivez une commande `sed` qui affiche `Alice - Bruxelles` sans modifier le fichier.

Exercice 7 (Lire, compléter et corriger du Bash)

(a) Sans exécuter le script, donnez la valeur finale de `somme` et l'affichage :

```
somme=0
for n in 1 4 6 2; do
    if (( n > 2 )); then
        somme=$((somme + n))
    fi
done
echo "$somme"
```

(b) Complétez une fonction `copie` telle que `copie notes.txt` crée `notes.txt.bak`.

```
copie() {
    fichier=__________
    cp ___________ ___________
}
```

(c) Le script suivant devrait afficher la somme des entiers de 1 à 4. Identifiez et corrigez ses deux erreurs.

```
total=0
for i in 1 2 3 4; do
    total=$((total - i))
done
echo "total"
```

Période 4 – Mini-examen blanc (sans terminal)

Consigne : travail individuel pendant 25 à 30 minutes. Utilisez uniquement les documents qui seront autorisés à l'examen. Répondez d'abord sans vérifier au terminal.

Exercice 8 (Questions rapides)

(a) À quoi servent `head`, `kill` et `apt` ?

(b) Depuis `/home/etudiant/travail/cours`, rejoignez `/home/etudiant/photos` par un chemin relatif.

(c) Écrivez une commande pour créer `travail.tar.gz` contenant le répertoire `travail`, puis une commande pour l'extraire.

(d) Interprétez `-rwxr-----` et donnez la commande `chmod` octale correspondante.

Exercice 9 (Manipuler du texte)

(a) Après `echo A > x.txt`, `echo B >> x.txt`, `echo C > x.txt`, que contient `x.txt` ?

(b) Écrivez une commande `grep -E` pour sélectionner dans `ids.txt` exactement deux chiffres suivis de quatre lettres majuscules.

(c) Dans `personnes.txt`, chaque ligne a la forme `nom:ville`. Écrivez une commande `sed` pour remplacer le séparateur par une virgule et un espace, sans modifier le fichier.

(d) Construisez un pipeline qui affiche les trois premières lignes distinctes, triées, du deuxième champ d'un fichier `donnees.txt` séparé par des deux-points.

Exercice 10 (Comprendre un script)

(a) Quel nombre est affiché ?

```
resultat=0
for x in 3 7 2 5; do
    if (( x < 6 )); then
        resultat=$((resultat + x))
    fi
done
echo "$resultat"
```

(b) Complétez cette fonction pour qu'elle copie le fichier reçu en argument en lui ajoutant `.old` :

```
conserver() {
    cp __________ __________
}
```

Pour aller plus loin (facultatif)

Exercice 11 (Défis)

(a) Écrivez une commande qui affiche les cinq lignes les plus fréquentes d'un fichier `evenements.txt`, avec leur nombre d'occurrences. Cherchez les options utiles de `uniq` et `sort`.

(b) Modifiez la fonction `copie` pour qu'elle refuse de remplacer un fichier `.bak` déjà existant.