FICHE 9 - Droits

1. `touch fichier.txt`
2. sachant que les droits par défaut lors de la création d'un fichier sont `rw-r--r--` (droits en écriture uniquement pour le propriétaire du fichier), les commandes suivantes sont équivalentes :

   `chmod -w fichier.txt`

   OU

   `chmod u-w fichier.txt`

   OU

   `chmod ugo-w fichier.txt`

   OU

   `chmod u=r fichier.txt`

   Remarque : lors de l'ajout (+) ou la suppression (-) d'un droit (rwx) sans spécifier le niveau (ugo), chmod modifie par défaut les permissions de tous les niveaux (ugo), à l'exception notable du droit de modification (w) qui ne change que les permissions de l'utilisateur car il s'agit du droit le plus potentiellement dangereux ; bash prévient d'ailleurs l'utilisateur de ce comportement, comme l'illustre l'exemple ci-dessous :

   ```
   anthony.legrand@courslinux:~$ ls -l fichier.txt
   --w--w--w- 1 anthony.legrand students 0 Feb 28 19:57 fichier.txt
   anthony.legrand@courslinux:~$ chmod -w fichier.txt
   chmod: fichier.txt: new permissions are ----w--w-, not ---------
   ```

3. `nano fichier.txt`

   → ne peut pas sauver (*permission denied*)

4. `mkdir test`

   `mkdir test/linux`

   `cp fichier.txt test`

5. `chmod -w test`

   `cd test`

   `mv fichier.txt fichier.doc`

   → pas le droit car fichiers du répertoire non ajoutables/supprimables (pas de droit w)

6. `cd ~`

   `chmod u+w test`

   `chmod –x test`

7. `cp fichier.txt test/linux`

   → pas le droit de traverser test/ (pas de droit x)

8. `chmod ugo+x test ; chmod ugo-r test`

   OU

   `chmod +x-r test`

   `cat test/fichier.txt`

   → OK (le chemin est valide car +x au repertoire test MAIS ls et complétion automatique KO)

9. `ls -ld test` (option –d : afficher les noms de répertoires eux-mêmes, pas leur contenu)

   `ls -l test`

   → pas le droit car contenu du répertoire non listable (pas de droit r)

10. `mkdir test2`

    `chmod 750 test2`

FICHE 10 - Filtres

1. `head -10 find.txt`

   OU

   `head find.txt` (car 10 lignes par défaut)

2. `tail -5 find.txt`
3. `tail -n +1201 find.txt` (-n +K : afficher à partir de la Kième ligne)[1]

   [pour vérifier : `tail -n +3001 find.txt | wc -l ; wc -l find.txt`]

4. `nl find.txt` (par défaut : ne numérote pas les lignes vides ⇔ `nl -b t find.txt` ;

   pour numéroter aussi les lignes vides : `nl -b a find.txt`)

5. `wc -l find.txt` (wc affiche : #lignes #mots #bytes)

   `wc -w find.txt`

   OU

   `wc -lw find.txt`

6. `sort find.txt` (ne tient compte que des lettres ; place les lignes vides au début)
7. `uniq find.txt` → KO : ne change rien car fichiers sans doublons successifs

   `cat find.txt | sort | uniq` → (pas vu mais OK : les lignes étant triées, uniq fonctionne!)

   (pour vérifier, compter les lignes résultantes en ajoutant : `| wc -l`)

8. `cat /etc/passwd`
9. `cut /etc/passwd -d: -f1` (-d définit le délimiteur ; -f définit une liste de champs/fields, ex : -f1,3)
10. `tr -d '\015' < DOS.txt > LINUX.txt` (-d pour éliminer/delete les caractères)

    OU

    `tr -d "\015" < DOS.txt > LINUX.txt`

    Attention : `tr` lit les lignes à l'entrée standard et affiche leur 'traduction' à la sortie standard.

    Pour tester ces commandes, téléchargez le fichier DOS.txt avec la commande :

    ```
    wget https://courslinux.ipl.be/~anthony.legrand/DOS.txt
    ```

    Exécutez la commande ci-dessus qui redirige le fichier DOS.txt vers l'entrée standard et la sortie standard vers le fichier LINUX.txt. Comparez ensuite le contenu des fichiers grâce au viewer hexadécimal `xxd` :

    ```
    xxd DOS.txt
    xxd LINUX.txt
    ```

[1] Notez que les propriétés de la fenêtre du terminal (nombre de colonnes, police de caractères définie) influenceront la taille d'une ligne et donc le nombre et le contenu des lignes affichées par une commande.