TP 1 : commandes shell et arborescence

Dans ce TP, nous allons apprendre quelques commandes shell (bash) pour :
- créer un répertoire ou un fichiers
- naviguer dans une arborescence
- déplacer ou renommer des fichiers
- comprendre la notion d'adresse absolue et relative

- Créer un dossier NSI dans son répertoire personnel

- Chercher dans les menus : Terminal
ou clic droit sur Bureau : nouveau Terminal

Listing :
```
ls
```
=> que fait cette commande ?

Nettoyer la console :
```
clear
```
=> que fait cette commande ?

Créer un répertoire (make directory)
```
mkdir commande-shell
ls
```
=> que fait la commande mkdir ?

Créer un fichier
```
touch question.txt
```
Chercher le fichier créé dans l'explorateur et l'ouvrir.
Saisir l'énoncé ci-dessous, enregistrer le fichier puis retourner dans le Terminal :
question 1 : indiquer un système d'exploitation libre
a- windows
b- macos

Afficher le contenu d'un fichier
```
cat question.txt
```

Editer un fichier
```
nano question.txt
```
Ajouter la réponse : *c- Linux*
et enregistrer
Ctrl X puis validation

Naviguer dans les répertoires (change directory)
```
cd commande-shell
```
=> que fait cette commande ?

Remonter d'un niveau d'arborescence :
```
cd ..
```
=> .. est le répertoire parent du répertoire courant (le répertoire courant est désigné par un seul .)

Exercice : Créer une arborescence :
Créer les répertoire et fichiers nécessaires pour obtenir l'arborescence suivante :

NSI
- **commande-shell**
- **cours**
  - chap1
  - chap2
  - chap3
- **exos**
  - **exo1**
    - q1
    - q2
  - **exo2**
    - q1
    - q210
- question.txt

La flèche <haut> rappelle les dernières commandes : très pratique lors d'opérations répétitives !

La commande tree permet de rapidement savoir où vous en êtes..

Déplacer un fichier (move)
Naviguer vers le répertoire NSI puis :

La touche tabulation <tab> appelle le service d'auto-complétion : gagne du temps et évite les fautes de frappe !

```
mv question.txt exos/exo1
cd exos/exo1
ls
```

=> qu'a fait la commande mv ?

Renommer un fichier (c'est une forme de déplacement : move) :

```
cd ../exo2
mv q210 q2
ls
```

=> qu'y a t il de nouveau ?

Supprimer un fichier

N'oubliez pas l'option -i : permet d'avoir un msg de confirmation avant la suppression !

```
rm -i q2
```

Exercice :
Sans quitter le répertoire NSI/exos/exo2, créer le répertoire NSI/exos/exo3/ puis déplacer le fichier question.txt dans ce répertoire NSI/exos/exo3/ en le renommant question.dat
L'arborescence doit alors être :
NSI
- commande-shell
- cours
  - chap1
  - chap2
  - chp3
- exos
  - exo1
    - q1
    - q2
  - exo2
    - q1
  - exo3
    - question.dat

Découvrir tous les secrets d'une commande et ses multiples options :

man *nom_commande*

Touche **q** pour quitter le manuel.

=> Recopier les 2 lignes de commandes saisies :

Compléments : glob, pipe, redirection de sortie

Le caractère * (glob) remplace n'importe quel caractère ou suite de caractères.
Ex : copie tous les fichiers png et jpg dans le dossier image contenu dans le répertoire courant.

```
cp *.png *.jpg  ./image
```

Ex : liste tous les fichiers commençant par class et finissant par l'extension .txt

```
ls class*.txt
```

Le caractère | (pipe) permet de diriger la sortie d'une commande vers l'entrée d'une autre commande.
Ex :
step 1) Génère une pensée ou une citation aléatoire

```
fortune
```

step 2) Génère en ASCII une vache qui dit Bonjour

```
cowsay 'Bonjour'
```

step 3) Renvoie la phrase générée par la commande fortune à la commande cowsay, donc une vache énonce une pensée ou une citation !

```
fortune | cowsay
```

On peut rediriger la sortie standard (la console) vers un fichier texte avec le symbole chevron >.
Ex : Créer le fichier message.txt qui contient une pensée (aucun retour en console) :

```
fortune > message.txt
```

Ex : Créer un fichier qui contient une vache exprimant une pensée :

```
fortune | cowsay > msg-de-vache.txt
```

Remarque : si le fichier existe déjà, son contenu est écrasé !
Pour ajouter du texte à la fin du fichier, il faut utiliser un double chevron >>.
Ex : ajouter une nouvelle pensée au fichier message.txt

```
fortune >> message.txt
```

TP 2 : gérer les droits (lecture, écriture, exécution)
read, write, execute

Dans ce TP, nous allons apprendre quelques commandes shell (bash) pour :
- lire les droits d'un répertoire ou d'un fichier
- modifier ces droits

Retour à la « maison » (HOME)

```
cd ~
```

=> le ~ est un abrégé pour désigner le répertoire personnel (home) de l'utilisateur.

Listing détaillé :

```
ls -l
```

=> on obtient des infos supplémentaires avec l'option -l ?
On trouve dans l'ordre :
- les **droits** sur les fichiers (on va s'y attarder plus tard)
- le **propriétaire** du fichier
- le **groupe** auquel appartient le propriétaire
- la taille du fichiers
- sa dernière date de modification
- et enfin le nom du fichier

Avec le système d'exploitation Linux, chaque utilisateur a un nom et appartient à un (ou plusieurs) groupe(s).
Les fichiers appartiennent à un utilisateur, et **différents droits sont accordés aux différents utilisateurs** sur chaque fichier (ou dossier) :
- r : droit de **lecture** (read) : permet de lire un fichier, voir son contenu…
- w:droit d'**écriture** (write) : permet de modifier le fichier
- x : droit d'**exécution** (execute) : permet d'exécuter le fichier

L'indication des droits est de la forme : - r w x - - x r - x :
- le 1er caractère indique si on parle d'un fichier (-) ou un répertoire (d, pour directory)
- les 3 caractères suivants sont les droits du propriétaire (r, w, x) (user : u)
- puis les 3 caractères suivants sont les droits des utilisateurs appartenant au groupe (group : g)
- les 3 derniers caractères sont les droits des autres utilisateurs (other : o)

Avec l'exemple - r w x - - x r - x : le fichier est accessible en lecture-écriture-exécution pour le propriétaire, seulement en exécution pour le groupe, et en lecture-exécution pour les autres.

Modification des droits (change mode) : Méthode 1

```
chmod [u,g,o] [+,-]  [r,w,x] [nom_du_fichier]
```

=> le choix u, g ou o est pour désigner les utilisateurs (user, group, other)
=> le choix + ou – est pour l'ajout ou le retrait de droits
=> le choix r, w ou x est la catégorie du droit

Exemple : La commande ci-dessous ajoute le droit d'écriture au groupe :

```
chmod g+w nom_du_fichier
```

Modification des droits (change mode) : Méthode 2

```
chmod [n1n2n3] [nom_du_fichier]
```

=> Les 3 nombres n_1, n_2 et n_3 décrivent en mode octal les permissions de u, g et o selon le principe suivant :
r w x : « correspondent » à un nombre binaire de 3 bits : donc x vaut 1, w vaut 2 et r vaut 4.
Le nombre n étant la somme des 3.

Exemples : r-x vaut 5, rwx vaut 7, --x vaut 1, etc...

Exemple : La commande ci-dessous donne tous les droits à tout le monde ! :

```
chmod 777 nom_du_fichier
```

Exercice : (tout doit être fait depuis un Terminal !)
1- Préliminaire :
- créer un fichier test_droits
- éditer le fichier et y ajouter un peu de texte, puis sauvegarder.
2- Relever les droits par défauts de ce fichier.
3- Ajouter les droits d'écriture et d'exécution au groupe.
4- Retirer le droit de lecture aux autres.
5- Avec la « méthode 2 » rétablir les droits qui existaient par défaut.
6- Retirer le droit d'écriture à l'utilisateur et essayer de le modifier avec l'éditeur nano.
7- Retirer le droit de lecture à l'utilisateur et essayer de le visualiser avec la commande less.