Commandes système sous Linux

Travaux pratiques sur le simulateur de terminal

Résumé

Ce document vous guide dans la découverte des commandes fondamentales d'un système Linux. Vous utiliserez le **simulateur de terminal** `https://terminal.edulatex.xyz` pour tester chaque commande. N'hésitez pas à expérimenter au-delà des exemples proposés.

① Navigation dans l'arborescence

Définition 1 **(Arborescence de fichiers)** Sous Linux, les fichiers et dossiers sont organisés en une structure hiérarchique appelée **arborescence**. La racine de cette arborescence est notée `/`. Chaque utilisateur dispose d'un répertoire personnel situé dans `/home/`.

Chemins absolus et relatifs. Un **chemin absolu** commence par `/` et décrit le trajet depuis la racine. Un **chemin relatif** décrit le trajet depuis le répertoire courant. Le symbole `~` désigne le répertoire personnel de l'utilisateur.

Exercice 1 *Découvrir l'arborescence*

1. Affichez le répertoire courant avec la commande `pwd`.
2. Listez le contenu du répertoire courant avec `ls`.
3. Déplacez-vous dans le répertoire `/home` avec `cd /home`, puis vérifiez avec `pwd`.
4. Revenez dans votre répertoire personnel avec `cd ~`.
5. Utilisez `ls -la` pour afficher les fichiers cachés. Que remarquez-vous ?

Exercice 2 *Manipuler les chemins*

1. Depuis votre répertoire personnel, affichez le contenu de `/etc` sans vous y déplacer avec `ls /etc`.
2. Créez un dossier `projets` avec `mkdir projets`.
3. Déplacez-vous dans `projets`, puis créez un sous-dossier `tp1` avec `mkdir tp1`.
4. Revenez au répertoire parent avec `cd ..` et vérifiez avec `pwd`.
5. Affichez l'arborescence complète de `projets` avec `tree projets`.

② Manipulation de fichiers et dossiers

À retenir. Les commandes de manipulation de fichiers sont destructives : une suppression avec `rm` est définitive (il n'y a pas de corbeille sous Linux). Soyez toujours prudent avant de supprimer un fichier ou un dossier.

Exercice 3 *Créer et supprimer*

1. Créez un fichier vide `notes.txt` avec `touch notes.txt`.
2. Écrivez du texte dedans avec `echo "Ceci est un test" > notes.txt`.
3. Vérifiez le contenu avec `cat notes.txt`.
4. Ajoutez une deuxième ligne avec `echo "Deuxième ligne" >> notes.txt`. Quelle est la différence entre `>` et `>>` ?
5. Copiez le fichier avec `cp notes.txt copie.txt`, puis vérifiez avec `ls`.
6. Renommez `copie.txt` en `sauvegarde.txt` avec `mv copie.txt sauvegarde.txt`.
7. Supprimez `sauvegarde.txt` avec `rm sauvegarde.txt`.

Exercice 4 *Opérations sur les dossiers*

1. Créez une arborescence en une seule commande avec `mkdir -p cours/nsi/chapitre1`.
2. Vérifiez le résultat avec `tree cours`.
3. Créez un fichier `cours/nsi/chapitre1/resume.txt` avec un contenu de votre choix.
4. Essayez de supprimer le dossier `cours` avec `rm cours`. Que se passe-t-il ?
5. Utilisez `rm -r cours` pour supprimer le dossier et tout son contenu.
6. Expliquez pourquoi l'option `-r` est nécessaire.

③ Traitement de texte

Information. La philosophie Unix repose sur des outils simples qui font une seule chose bien, et que l'on combine avec le **pipe** (`|`) pour réaliser des traitements complexes. Le pipe transmet la sortie d'une commande comme entrée de la suivante.

Exercice 5 *Filtrage et recherche* Commencez par créer un fichier de données. La commande `printf` écrit le texte en remplaçant chaque `\n` par un retour à la ligne (la commande `echo` de Bash, elle, écrirait `\n` tel quel) :

```
printf "Alice 15\nBob 12\nCharlie 18\nDiana 15\nEve 12\nFrank 20\n" > eleves.txt
```

1. Affichez les 3 premières lignes avec `head -n 3 eleves.txt`.
2. Affichez les 2 dernières lignes avec `tail -n 2 eleves.txt`.
3. Comptez le nombre de lignes avec `wc -l eleves.txt`.
4. Recherchez les élèves ayant 15 avec `grep 15 eleves.txt`.
5. Triez le fichier par ordre alphabétique avec `sort eleves.txt`.

6. Combinez les commandes : `cat eleves.txt | sort | uniq`. Expliquez le résultat.
7. Recherchez les lignes ne contenant pas "15" avec `grep -v 15 eleves.txt`.

Exercice 6 *Pipes et enchaînements*

1. Créez un fichier `fruits.txt` contenant : pomme, banane, cerise, pomme, banane, datte, pomme.
2. Comptez le nombre d'occurrences de chaque fruit avec `sort fruits.txt | uniq -c`.
3. Affichez uniquement les lignes contenant "an" avec `grep an fruits.txt`.
4. Comptez le nombre de lignes contenant "pomme" avec `grep -c pomme fruits.txt`.
5. Recherchez récursivement le mot "test" dans tous les fichiers du répertoire courant avec `grep -r test .`
6. Enchaînez deux commandes avec `&&` : créez un dossier et déplacez-vous dedans en une seule ligne.

4 Permissions et utilisateurs

Définition 2 **(Permissions Unix)** Chaque fichier possède trois catégories de permissions : **lecture** (`r`), **écriture** (`w`) et **exécution** (`x`), définies pour trois entités : le **propriétaire** (`u`), le **groupe** (`g`) et les **autres** (`o`). Ces permissions sont affichées sous la forme `rwxrwxrwx` ou en notation octale (par exemple `755`).

Notation octale. En notation octale, chaque chiffre représente les permissions d'une entité : `r`=4, `w`=2, `x`=1. Par exemple, `755` signifie : propriétaire = 7 (rwx), groupe = 5 (r-x), autres = 5 (r-x).

Exercice 7 *Gérer les permissions*

1. Créez un fichier `script.sh` avec `echo '#!/bin/bash' > script.sh` (avec des guillemets doubles, Bash interpréterait le `!` et afficherait une erreur).
2. Affichez ses permissions avec `ls -l script.sh`. Décrivez-les.
3. Rendez le fichier exécutable pour le propriétaire avec `chmod u+x script.sh`.
4. Vérifiez le changement avec `ls -l script.sh`.
5. Utilisez la notation octale pour donner les permissions `rwxr-xr--` avec `chmod 754 script.sh`.
6. Vérifiez avec `stat script.sh` les informations détaillées du fichier.

Exercice 8 *Utilisateurs et droits*

1. Affichez votre nom d'utilisateur avec `whoami`.
2. Affichez vos identifiants avec `id`. Que représentent l'UID et le GID ?
3. Essayez de passer en superutilisateur avec `su root`. Que se passe-t-il ?
4. Utilisez `sudo ls /root`. Quelle est la différence entre `su` et `sudo` ?

5 Informations système

Exercice 9 *Explorer le système*

1. Affichez le nom de la machine avec `hostname`.
2. Affichez les informations du système avec `uname -a`. Identifiez le noyau et l'architecture.
3. Affichez la date et l'heure avec `date`.
4. Affichez l'utilisation du disque avec `df -h`. Que signifie l'option `-h` ?
5. Affichez l'utilisation de la mémoire avec `free -h`.

Exercice 10 *Historique et variables*

1. Tapez plusieurs commandes, puis affichez l'historique avec `history`.
2. Affichez les variables d'environnement avec `env`. Identifiez la variable `HOME`.
3. Affichez la valeur de `HOME` avec `echo $HOME`.
4. Créez une variable avec `export MA_VAR=bonjour`, puis affichez-la avec `echo $MA_VAR`.
5. Créez un alias avec `alias ll="ls -la"`, puis testez-le.

6 Recherche de fichiers

Exercice 11 *Trouver et identifier*

1. Créez l'arborescence suivante :

```
mkdir -p projet/src projet/docs projet/tests
touch projet/src/main.py projet/src/utils.py
touch projet/docs/readme.txt projet/tests/test_main.py
```

2. Recherchez tous les fichiers Python avec `find projet -name "*.py"`.
3. Recherchez uniquement les dossiers avec `find projet -type d`.
4. Identifiez le type du fichier `projet/src/main.py` avec `file projet/src/main.py`.
5. Vérifiez l'emplacement de la commande `ls` avec `which ls`.
6. Consultez le manuel de la commande `grep` avec `man grep`.

Résumé. **Commandes essentielles à retenir :**

— **Navigation :** `pwd`, `cd`, `ls`, `tree` ;
— **Fichiers :** `touch`, `cp`, `mv`, `rm`, `mkdir`, `cat`, `echo` ;
— **Texte :** `head`, `tail`, `wc`, `grep`, `sort`, `uniq` ;
— **Permissions :** `chmod`, `ls -l`, `whoami`, `id`, `sudo` ;
— **Système :** `uname`, `hostname`, `date`, `df`, `free`, `history` ;
— **Recherche :** `find`, `file`, `which`, `man` ;
— **Opérateurs :** `>` (redirection), `>>` (ajout), `|` (pipe), `&&` (enchaînement).