NOM: PRÉNOM: CLASSE:

MR ROLIN 2025

[DEV] Commandes et scripts

Objectifs

- Comprendre ce qu'est une ligne de commande (CLI) et son rôle.
- Apprendre les commandes de base pour naviguer, gérer des fichiers et dossiers.
- Comprendre ce qu'est un script et savoir en écrire un simple.
- Être capable d'automatiser une petite tâche avec un script.

Introduction

La ligne de commande (CLI) est un mode d'interaction avec l'ordinateur où l'utilisateur tape des instructions textuelles. Elle s'oppose à l'interface graphique (GUI), où l'on utilise des clics de souris et des menus.

Avantages de la CLI : rapidité, possibilité d'automatiser des tâches répétitives, faible consommation de ressources, universalité (Linux, Windows, macOS partagent une logique commune).

Inconvénients : peu intuitive pour les débutants, nécessite de mémoriser les commandes, erreurs pouvant être critiques (ex : suppression accidentelle de fichiers), pas d'éléments visuels comme des barres de progression.

Exemples concrets d'utilisation :
- Un administrateur système peut créer des comptes utilisateurs rapidement.
- Un développeur peut compiler et tester son code plus vite qu'avec une interface graphique.
- Un technicien peut réparer une machine même quand l'interface graphique ne démarre plus.

Structure d'une commande

Une commande est composée de trois parties :
- **Nom de la commande** : l'action de base (ex. ls, dir, mkdir).
- **Options/paramètres** : modifient le comportement de la commande. Sur Linux/macOS elles commencent par - (ex. -l) ou -- (ex. --all), sur Windows par / (ex. /s).
- **Arguments** : la ou les cibles (fichier, dossier, programme).

Exemples :
- Unix : `ls -l /home/user` → liste le contenu du dossier /home/user en affichage détaillé.
- Windows : `dir C:\Users /s` → affiche aussi le contenu des sous-dossiers.

Navigation dans le système de fichiers

Un système de fichiers est organisé en arborescence : dossiers et sous-dossiers contiennent des fichiers.

Racine : `/` (Linux/macOS) ou `C:\` (Windows).

Répertoires spéciaux :
- `.` → dossier courant (là où vous êtes)
- `..` → dossier parent (remonter d'un niveau)
- `~` (Linux/macOS) → dossier personnel de l'utilisateur (/home/utilisateur)
- `/` (Linux) ou `C:\` (Windows) → racine du disque

Variables d'environnement : `$HOME` (Unix) ou `%USERPROFILE%` (Windows)

Action	Unix	Windows

Afficher le dossier courant	`pwd`	`cd`
Entrer dans un dossier	`cd dossier`	`cd dossier`
Remonter d'un niveau	`cd ..`	`cd ..`
Lister le contenu	`ls`	`dir`
Afficher l'arborescence	`tree`	`tree`

Manipulation de fichiers et dossiers

Avec la CLI, on peut gérer directement des fichiers et dossiers. Attention : certaines commandes sont destructives et irréversibles.

Action	Unix	Windows
Créer un dossier	`mkdir dossier`	`mkdir dossier`
Créer un fichier texte	`touch fichier.txt`	`echo Bonjour > fichier.txt`
Lire un fichier	`cat fichier.txt`	`type fichier.txt`
Supprimer un fichier	`rm fichier.txt`	`del fichier.txt`
Supprimer un dossier	`rm -r dossier`	`rmdir /s dossier`
Copier un fichier	`cp fichier.txt copie.txt`	`copy fichier.txt copie.txt`
Déplacer/renommer un fichier	`mv fichier.txt nouveau.txt`	`move fichier.txt nouveau.txt`

⚠️ Prudence : la commande rm -r sous Unix/Linux peut supprimer tout un dossier avec son contenu sans demander confirmation.

Redirections et opérateurs

La CLI permet de rediriger la sortie ou l'entrée d'une commande.

Action	Unix	Windows
Rediriger sortie (écrase)	`ls > liste.txt`	`dir > liste.txt`
Ajouter sortie (sans écraser)	`echo Ligne >> notes.txt`	`echo Ligne >> notes.txt`
Lire depuis fichier (entrée)	`sort < noms.txt`	`sort < noms.txt`
Pipe (chaîner commandes)	`ls | grep ".txt"`	`dir | findstr ".txt"`
Conditionnel (si succès) → la 2e commande ne s'exécute que si la 1re réussit	`commande1 && commande2`	`commande1 && commande2`

Conditionnel (si échec) → la 2e commande s'exécute seulement si la 1re échoue	`commande1 || commande2`	`commande1 || commande2`

Sous Windows, les opérateurs `&&` et `||` fonctionnent directement dans cmd.exe et dans PowerShell.

Recherche

La recherche permet de localiser rapidement un fichier ou de retrouver du texte précis à l'intérieur d'un document sans avoir à parcourir manuellement tout le contenu.

Action	Unix	Windows
Rechercher un fichier	`find / -name fichier.txt`	`dir /s fichier.txt`
Rechercher texte dans fichier	`grep mot fichier.txt`	`findstr mot fichier.txt`
Options utiles	`grep -i` (ignore casse) `grep -n` (numéro lignes)	`findstr /i` `findstr /n`

Scripts

Un script est un fichier texte qui contient une suite de commandes. Il permet d'automatiser des tâches que l'on ferait manuellement.

Intérêts des scripts :

- Automatiser des sauvegardes quotidiennes.
- Lancer plusieurs programmes d'un coup.
- Réaliser des déploiements sur plusieurs machines.
- Réduire les erreurs humaines grâce à la répétition exacte.

Unix

```
#!/bin/bash
echo "Bonjour"
mkdir projet
ls -l
```

Windows

```
@echo off
echo Bonjour
mkdir projet
dir
```

Exécution :

- Unix : donner les droits (`chmod +x script.sh`) puis `./script.sh`.
- Windows : double-cliquer sur le .bat ou lancer dans l'invite de commande.

Structure de script

Commentaires : Ligne qui explique le fonctionnement sans être exécutée.

- Unix : # → `# Script de sauvegarde`
- Windows : REM → `REM Script de sauvegarde`

Variables : stocker des valeurs à réutiliser.

- Unix : `NOM="Axel" ; echo "Bonjour $NOM"`

- Windows : `set NOM=Axel & echo Bonjour %NOM%`

Conditions : exécuter des commandes seulement si une condition est remplie.

- Unix :
```
if [ -f fichier.txt ]; then
  echo "Le fichier existe"
else
  echo "Le fichier n’existe pas"
fi
```
- Windows :
```
if exist fichier.txt (
  echo Le fichier existe
) else (
  echo Le fichier n’existe pas
)
```

Boucles : répéter une action pour plusieurs fichiers.

- Unix :
```
for fichier in *.txt; do
  echo "Traitement de $fichier"
done
```
- Windows :
```
for %%f in (*.txt) do (
  echo Traitement de %%f
)
```

Exemples

Unix

```
#!/bin/bash
# Crée une petite arbo, manipule des fichiers, filtre, sauvegarde.

# Crée les dossiers TP/Docs et TP/Backup (sans erreur si déjà là)
mkdir -p ~/TP/Docs ~/TP/Backup
# Va dans TP/Docs ou quitte si échec
cd ~/TP/Docs || exit 1

# Crée note1.txt avec le texte "Bonjour"
echo "Bonjour" > note1.txt
# Crée note2.txt avec le texte "Informatique"
echo "Informatique" > note2.txt

# Liste le contenu du dossier dans liste.txt
ls > liste.txt
# Garde seulement les lignes contenant ".txt" dans juste_txt.txt
cat liste.txt | grep ".txt" > juste_txt.txt

# Copie note1.txt vers le dossier Backup
cp note1.txt ../Backup/
# Renomme note2.txt en note2_old.txt
mv note2.txt note2_old.txt
```

Windows

```
@echo off
REM Crée une petite arbo, manipule des fichiers, filtre, sauvegarde.

mkdir "%USERPROFILE%\TP\Docs" "%USERPROFILE%\TP\Backup" 2>nul
cd /d "%USERPROFILE%\TP\Docs" || goto :eof

REM Fichiers de test
echo Bonjour>note1.txt
echo Informatique>note2.txt

REM Listing et filtre
dir /b > liste.txt
type liste.txt | findstr ".txt" > juste_txt.txt

REM Copie et renommage
copy /y note1.txt "%USERPROFILE%\TP\Backup\" >nul
ren note2.txt note2_old.txt
```