# Introduction à l'invite de commande

## 1. **Introduction**

L'invite de commande (aussi appelée **terminal**, **console**, ou **CLI** pour *Command Line Interface*) est un outil qui permet de communiquer avec l'ordinateur en tapant des **commandes** textuelles, plutôt que d'utiliser une interface graphique (fenêtres, icônes, menus).

Son utilisation :

- Automatiser des tâches (grâce à des scripts).
- **Indispensable** dans de nombreux domaines : programmation, développement web, DevOps, cybersécurité, intelligence artificielle, etc.
- Certains outils **ne fonctionnent que via la ligne de commande** (Git, Docker, etc.).
- **Plus d'options** que les interfaces graphiques.
- **Universelle** : ce que vous apprenez ici fonctionne sur tous les systèmes (Windows, Linux, macOS) avec quelques différences.

---

### Exemple simple :

Quand on utilise une interface graphique pour aller dans un dossier :

- On ouvre l'explorateur
- On clique plusieurs fois...

Avec la ligne de commande :

```
cd Documents/Projets/Exemple
```

→ Une seule ligne suffit.

---

## 2. **Premiers pas dans l'invite de commande**

- Ouvrir l'invite de commande :
    - Windows : `cmd`, `PowerShell`, ou `Windows Terminal`
    - macOS / Linux : Terminal

**Structure d'une commande**

Chaque commande suit en général cette **structure** :

```
commande [options] [arguments]
```

- **commande** : Le nom du programme ou outil (ex : `ls`, `cd`, `mkdir`)
- **option(s)** : Paramètres précédés d'un $-$ ou `--` pour modifier le comportement
- **argument(s)** : Ce sur quoi la commande agit (ex : un nom de fichier)

Exemple :

```
ls -l /home/user/Documents
```

- `ls` = liste les fichiers
- $-$`l` = option pour afficher plus de détails
- `/home/user/Documents` = chemin du dossier à afficher

**Règles importantes :**

| Règle | Explication |
|---|---|
| Les commandes sont **sensibles à la casse** | `ls` ≠ `LS` |
| L'espace sépare les **éléments** | `ls -l` (pas `ls-l`) |
| On peut **combiner des options** | `ls -la` = `ls -l -a` |
| Utiliser `Tab` pour l'autocomplétion | Évite les fautes de frappe |
| Utiliser les flèches ↑ ↓ | Pour naviguer dans l'historique des commandes |

---

## 3. **Navigation dans le système de fichiers**

### Objectif

Apprendre à se déplacer dans les dossiers du système via l'invite de commandes (`cmd`) sous Windows.

- `dir` – Afficher le répertoire courant
- `cd` – Changer de répertoire (Change Directory)
    - `cd ..`, `cd ~`, `cd /path/to/folder`
- `ls` ou `dir` – Lister les fichiers
    - Options : `ls -l`, `ls -a`, `ls -lh`

| Action | Commande | Exemple |
|---|---|---|
| Afficher le dossier courant | `cd` | `cd` |
| Lister les fichiers d'un dossier | `dir` | `dir` |
| Aller dans un sous-dossier | `cd nom_du_dossier` | `cd Documents` |
| Remonter d'un dossier | `cd ..` | `cd ..` |

| Action | Commande | Exemple |
| --- | --- | --- |
| Aller à la racine du disque | `cd \` | `cd \` |
| Aller à un chemin complet | `cd chemin_complet` | `cd C:\Users\Nom\Bureau` |
| Effacer l'écran | `cls` | `cls` |

## Exemples pratiques

### 1. Afficher le dossier courant

```
cd
```

Affiche le chemin du dossier dans lequel vous vous trouvez.

---

### 2. Voir le contenu du dossier

```
dir
```

Affiche la liste des fichiers et sous-dossiers.

---

### 3. Aller dans un dossier

```
cd Bureau
```

Entre dans le dossier "Bureau" (s'il existe dans le dossier courant).

> Utilisez `Tab` pour compléter automatiquement le nom du dossier.

---

### 4. Remonter d'un niveau

```
cd ..
```

Revient dans le dossier parent.

---

### 5. Aller directement dans un chemin complet

```
cd C:\Users\Etudiant\Documents
```

---

### Remarques importantes

- Sous `cmd`, **les chemins utilisent les antislashs (`\`)**.
- Sous PowerShell, les deux `\` et `/` fonctionnent en général.
- Si un nom de dossier contient des **espaces**, on l'entoure de guillemets :

```
cd "Mes Documents"
```

---

### Exercice

1. Ouvrir une invite de commande
2. Aller dans le dossier `Documents`
3. Créer un nouveau dossier depuis l'explorateur (`TestCLI`)
4. Utiliser `cd` pour entrer dans `TestCLI`
5. Taper `cd ..` pour revenir à `Documents`
6. Taper `cls` pour effacer l'écran

---

## 4. Gestion des fichiers

### Objectif

Apprendre à créer, supprimer, renommer, copier et déplacer des **fichiers** et **dossiers** en ligne de commande (`cmd`) sous Windows.

### Commandes de base

#### Créer un dossier (Make Directory)

```
mkdir nom_du_dossier
```

Crée un dossier vide.

---

#### Créer un fichier vide

```
type nul > nom_du_fichier.txt
```

Crée un fichier vide.

---

**Supprimer un fichier (Delete)**

```
del nom_du_fichier
```

---

**Supprimer un dossier**

```
rmdir nom_du_dossier
```

Pour supprimer un dossier et tout son contenu :

```
rmdir /s /q nom_du_dossier
```

---

**Renommer un fichier ou un dossier (Rename)**

```
ren ancien_nom nouveau_nom
```

**Exemples :**

```
ren notes.txt todo.txt
ren DossierA DossierB
```

---

**Copier un fichier**

```
copy source destination
```

**Exemple :**

```
copy notes.txt C:\Users\Etudiant\Bureau\
```

---

**Déplacer (ou renommer) un fichier**

```
move source destination
```

**Exemple :**

```
move notes.txt C:\Users\Etudiant\Documents\
```

---

Astuces utiles

| Astuce | Exemple |
|---|---|
| Créer plusieurs dossiers | `mkdir dossier1 dossier2` |
| Supprimer plusieurs fichiers | `del fichier1.txt fichier2.txt` |
| Utiliser `*` | `del *.txt` → supprime tous les fichiers `.txt` |
| Espaces dans les noms | `mkdir "Mon Dossier"` |

---

# 5. Lecture de fichiers, redirections et pipes

Objectif

Savoir **lire le contenu d'un fichier texte**, **rediriger l'affichage vers un fichier**, et **chaîner des commandes (pipes)**.

---

## A. **Lire le contenu d'un fichier**

`type`

Affiche le contenu d'un fichier texte dans le terminal.

```
type nom_du_fichier.txt
```

> Attention : affiche tout d'un coup, même pour de gros fichiers.

---

## B. **Redirections de sortie**

### Rediriger la sortie d'une commande dans un fichier

**`>` : redirection (écrase le fichier si existant)**

```
echo Bonjour > message.txt
```

Crée (ou écrase) `message.txt` et y écrit "Bonjour"

**`>>` : redirection (ajoute à la fin du fichier)**

```
echo Encore une ligne >> message.txt
```

Ajoute la ligne sans effacer ce qui existe

---

## C. **Utiliser les "pipes" (|)**

Le pipe permet de **chaîner deux commandes** : la sortie de la première devient l'entrée de la seconde.

```
commande1 | commande2
```

Exemple :

```
type message.txt | more
```

Affiche le contenu page par page (utile pour les longs fichiers)

---

Exemple avec `find` (recherche)

```
type message.txt | find "Bonjour"
```

Affiche uniquement les lignes contenant le mot "Bonjour"

---

Résumé

| Action | Commande |
|---|---|
| Lire un fichier | `type fichier.txt` |
| Écrire dans un fichier | `echo texte > fichier.txt` |
| Ajouter à un fichier | `echo texte >> fichier.txt` |
| Lire un fichier par pages | `type fichier.txt \| more` |
| Chercher un mot dans un fichier | `type fichier.txt \| find "mot"` |
| Trier le contenu | `sort < fichier.txt` |

# 6. Notions utiles pour bien utiliser l'invite de commande

## A. **Chemins absolus vs relatifs**

Comprendre les chemins est **essentiel** pour naviguer et manipuler des fichiers correctement.

### Chemin absolu

Chemin **complet** depuis la racine du disque.

```
cd C:\Users\Etudiant\Documents\Cours
```

Le chemin commence par une **lettre de disque** (ex. `C:`).

---

### Chemin relatif

Chemin **par rapport au dossier actuel**.

```
cd Cours\TP1
```

Fonctionne si vous êtes déjà dans `Documents`.

---

**Commande utile pour s'y retrouver :**

```
cd
```

Affiche le **chemin courant**.

---

## B. **Les dossiers spéciaux**

| **Dossier** | **Raccourci en `cmd`** |
|---|---|
| Dossier parent | `..` |
| Dossier courant | `.` |
| Racine du disque courant | `\` |
| Bureau (chemin absolu) | `C:\Users\NomUtilisateur\Desktop` |

Exemple :

```
cd ..
```

→ Remonte d'un dossier.

---

## C. **Fichiers cachés**

Sous Windows, certains fichiers sont **"cachés"** et ne s'affichent pas avec `dir` par défaut.

**Pour les voir :**

```
dir /a
```

| Option | Signification |
|---|---|
| `/a` | Affiche **tous** les fichiers |
| `/ah` | Affiche **seulement** les fichiers cachés |

> Les fichiers cachés ont l'attribut `H` dans la colonne d'attributs.

---

## D. **Attributs et permissions**

Sous Windows, il existe des **attributs de fichiers** (lecture seule, caché, etc.) qu'on peut modifier avec la commande `attrib`.

**Exemple :**

```
attrib +h fichier.txt
```

Rend le fichier **caché**.

```
attrib -h fichier.txt
```

Rend le fichier **visible**.

---

## Exercice

1. Naviguer jusqu'à le dossier `Documents`
2. Créer un fichier nommé `secret.txt`
3. Le rendre **caché** avec `attrib +h secret.txt`
4. Utiliser `dir` puis `dir /a` pour voir la différence

5. Le réafficher avec `attrib -h secret.txt`

---

## 7. Exercices pratiques

> Ces exercices et TP sont conçus pour un environnement **Windows**, dans l'**invite de commandes (`cmd`)**. Vous pouvez vous aider du **tableau des commandes** fourni pour réaliser les différentes actions.

---

Objectif

Mettre en pratique les notions vues en cours : navigation, création, manipulation de fichiers, redirections, affichage, pipes, copie, suppression et gestion des fichiers cachés.

---

### Exercice 1 – Création et navigation

1. Ouvrir l'invite de commande.
2. Aller sur le Bureau.
3. Créer un dossier nommé `TP_CLI`.
4. Entrer dans ce dossier.
5. Créer un sous-dossier `Notes`.

```
Bureau
└── TP_CLI
    └── Notes
```

---

### Exercice 2 – Fichiers simples

1. Créer un fichier vide nommé `todo.txt` dans `TP_CLI`.
2. Écrire une première ligne dans ce fichier : → *"Réviser la ligne de commande"*
3. Ajouter une deuxième ligne : → *"Faire les exercices"*
4. Afficher le contenu du fichier à l'écran.

```
Bureau
└── TP_CLI
    ├── Notes
    └── todo.txt
```

---

### Exercice 3 – Redirections et affichage

1. Créer un fichier `liste.txt` contenant 5 lignes de texte. → *Certaines lignes doivent contenir le mot « ligne », d'autres non.*
2. Afficher uniquement les lignes contenant le mot "ligne".

3. Afficher le contenu du fichier **page par page**.

```
Bureau
└── TP_CLI
    ├── Notes
    ├── todo.txt
    └── liste.txt
```

---

## Exercice 4 – Copie, déplacement et suppression

1. Copier le fichier `todo.txt` dans le dossier `Notes`.
2. Renommer `todo.txt` (dans `TP_CLI`) en `taches.txt`.
3. Supprimer le fichier `liste.txt`.
4. Supprimer le dossier `Notes`, **même s'il contient un fichier**.

```
Bureau
└── TP_CLI
    └── taches.txt
```

---

## Exercice 5 – Fichier caché

1. Créer un fichier nommé `secret.txt`.
2. Le rendre **caché** à l'aide de la commande appropriée.
3. Vérifier qu'il n'apparaît plus avec `dir`.
4. Le réafficher (le rendre visible de nouveau).

```
Bureau
└── TP_CLI
    └── secret.txt (caché)
```

---

### TP de synthèse

**Objectif** : Mettre en œuvre une séquence complète en autonomie, en mobilisant toutes les compétences vues jusqu'ici.

---

1. Sur le Bureau, créer un dossier `Projet_CLI`.

2. Dans ce dossier, créer deux sous-dossiers : `Docs` et `Sources`.

```
Bureau
└── Projet_CLI
```

```
    ├── Docs
    └── Sources
```

3. Créer un fichier `plan.txt` dans `Docs` contenant 3 lignes décrivant un projet fictif.
4. Copier ce fichier dans `Sources`.

```
Bureau
└── Projet_CLI
    ├── Docs
    │   └── plan.txt
    └── Sources
    |   └── plan.txt
```

5. Renommer `plan.txt` en `plan_final.txt` dans `Sources`.
6. Supprimer le fichier original dans `Docs`.

```
Bureau
└── Projet_CLI
    ├── Docs
    └── Sources
        └── plan_final.txt
```

7. Afficher le contenu de `plan_final.txt` **page par page**.
8. Rendre ce fichier **caché**.
9. Vérifier qu'il a disparu de l'affichage, puis le réafficher (le rendre visible).

```
Bureau
└── Projet_CLI
    ├── Docs
    └── Sources
        └── plan_final.txt (fichier caché)
```

---