BTS CIEL INFORMATIQUE ET RÉSEAUX

Les Scripts Linux - Partie 4

Entrées, Sorties et Fichiers

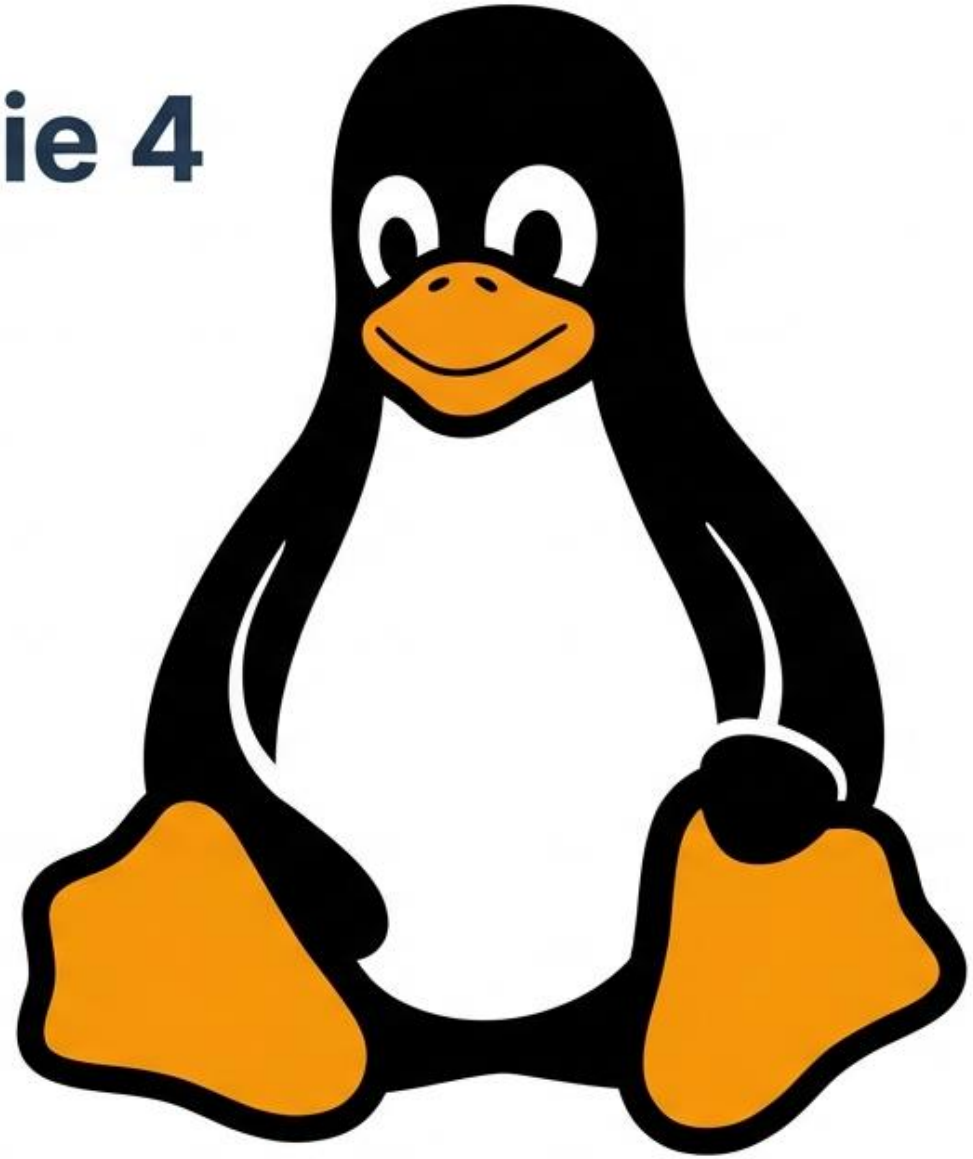
- Contexte : Maîtrise des interactions et de la logique conditionnelle dans le shell.
- Objectifs de la séance :

  - Gérer les interactions utilisateur (commandes read et echo)

  - Manipuler les flux de données (redirections et pipes)

  - Effectuer des tests logiques sur les fichiers (existence, type, droits)

  - Mise en pratique immédiate (4 exercices de scripting)



Lire les entrées utilisateur

La commande `read`

Définition :

`read` est utilisé pour lire une ligne de texte à partir de l'entrée standard (généralement, le clavier).

⏱️ **Note** : La commande suspend l'exécution du script jusqu'à ce que l'utilisateur valide avec Entrée.

Syntaxe :

```
read variable
```

```
echo "Quel est votre nom ?"
read nom
echo "Bonjour, $nom !"

Quel est votre nom ?
Alice
Bonjour, Alice !
```

Afficher des messages

La commande echo

Définition :

echo est utilisé pour afficher du texte sur la sortie standard (généralement, l'écran).

```
echo "Ceci est un test"


echo -e "Ligne 1\nLigne 2"
```

Options :

-n	Ne pas ajouter de nouvelle ligne à la fin (le curseur reste sur la même ligne).
-e	Activer l'interprétation des caractères échappés (indispensable pour utiliser \n pour un saut de ligne).

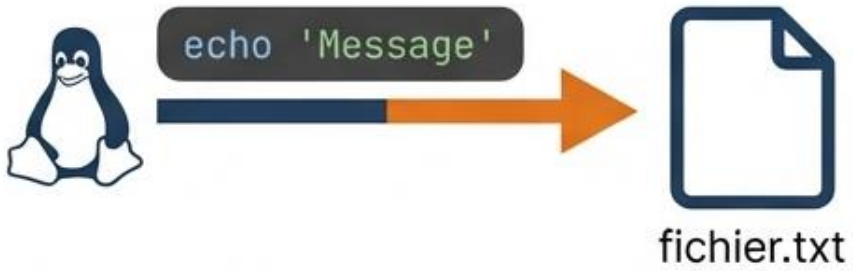
Astuce :

Sans l'option -e, le symbole \n est affiché comme du texte littéral.

Rediriger les flux (I/O)

Manipulation des sorties vers les fichiers

>
Écriture

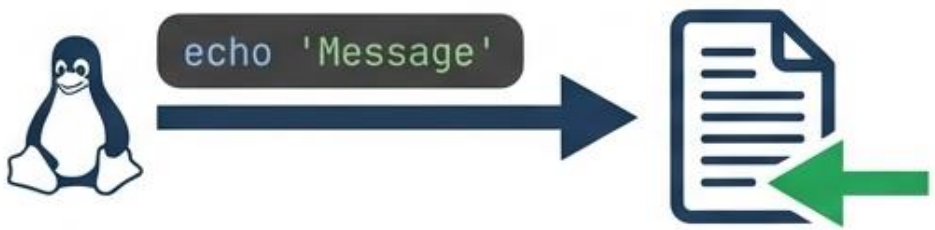


Redirige la sortie standard vers un fichier.

⚠ **Attention** : Écrase le fichier s'il existe déjà.

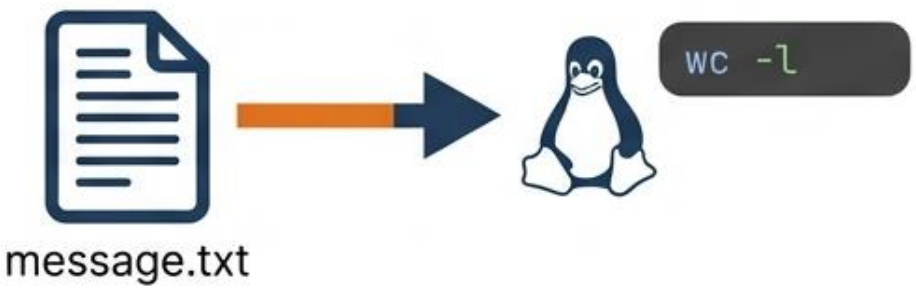
```
echo "Message" > fichier.txt
```

>>
Ajout



Redirige la sortie standard vers un fichier, en ajoutant le contenu à la fin (sans écraser).

<
Lecture



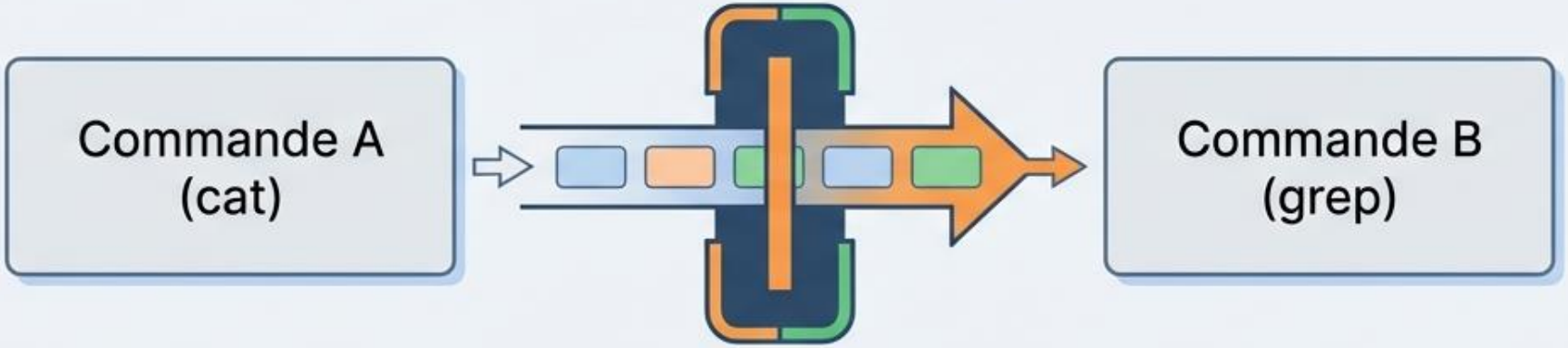
Lit l'entrée standard à partir d'un fichier.

```
wc -l < message.txt
```

Enchaîner les commandes

Les Pipes (|)

Le **pipe** est utilisé pour passer la sortie (output) d'une commande comme **entrée** (input) à une autre commande.




```
cat fichier.txt | grep "message"
```

Explication : On lit le fichier, puis on cherche le mot "message" dans ce flux.

Tests sur les fichiers (1/2)

Vérifier l'existence et l'état

Syntaxe : `[ option $FICHIER ]`

-e

Existe

Vrai si l'objet existe.

-s

Non vide

Vrai si l'objet existe et que sa taille est supérieure à zéro.

-f

Fichier Standard

Vrai si c'est un fichier (pas un dossier).

-r

Lisible

Vrai si le fichier possède la permission de lecture.

Tests sur les fichiers (2/2)

Vérifier les permissions et le type

-w

Inscriptible

Vrai si le fichier est modifiable (permission d'écriture).

-x

Exécutable

Vrai si le fichier peut être exécuté (permission d'exécution).

-d

Répertoire

Vrai si l'objet est un dossier/répertoire.

Note : Ces tests sont généralement utilisés à l'intérieur d'une boucle if.

Travaux Pratiques - Scripting

Mise en application des concepts

Consignes : Réalisez les scripts demandés en combinant les notions vues précédemment :

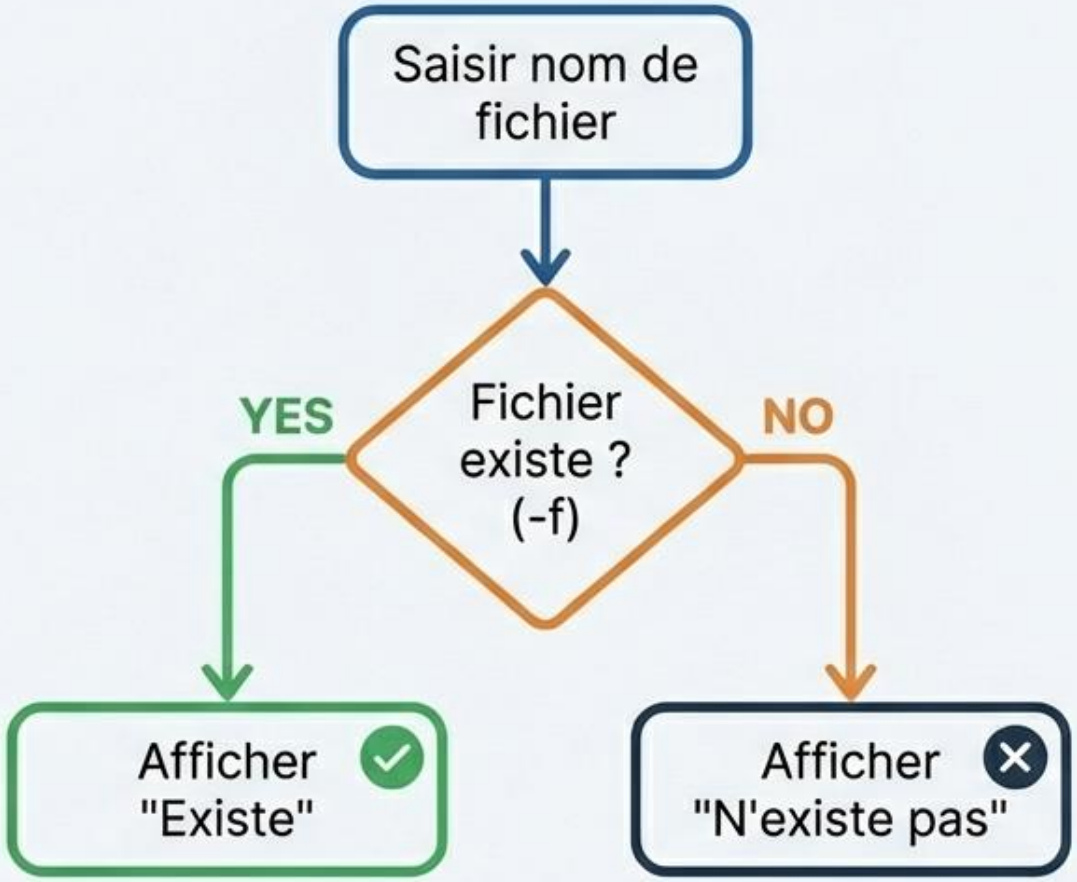
- 1. Interaction (read, echo)
- 2. Logique conditionnelle (if, tests [])
- 3. Flux de données (Redirections)

Exercice 1 : Vérifier l'Existence

Objectif : Écrire un script qui demande un nom de fichier et vérifie sa présence.

Algorithme à implémenter :

- 1. **Demander** : Utiliser `read` pour obtenir le nom du fichier.
- 2. **Tester** : Utiliser une instruction `if` avec le test `-f`.
- 3. **Informer** : Afficher un message "Le fichier existe" ou "Le fichier n'existe pas".

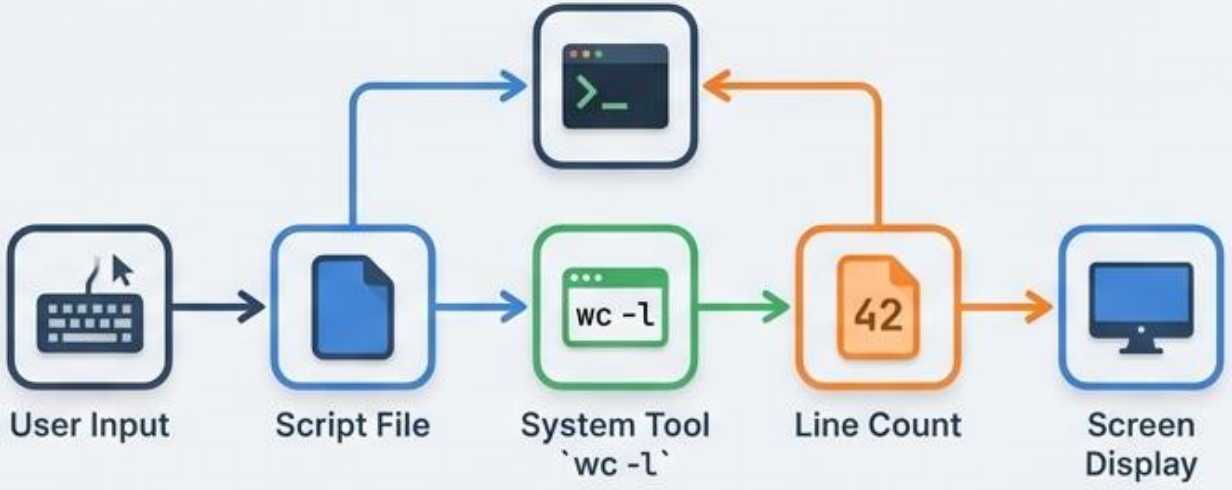


Exercice 2 : Compter les Lignes

Objectif : Créer un script qui lit un nom de fichier et affiche son nombre de lignes.

Algorithme à implémenter :

1. **Saisie** : Demander à l'utilisateur de saisir le nom d'un fichier.
2. **Traitement** : Exécuter la commande `wc -l` sur ce fichier.
3. **Affichage** : Présenter le résultat proprement à l'écran.



Exercice 3 : Copier un Fichier

Objectif : Script pour copier un fichier d'une source vers une destination.

Algorithme à implémenter :

1. **Saisie multiple** : Utiliser `read` pour obtenir le nom 'Source' et le nom 'Destination'.
2. **Action** : Utiliser la commande `cp` pour effectuer la copie.
3. **Vérification** : Vérifier si la copie a réussi (optionnel : via code de retour ou test).
4. **Feedback** : Afficher un message de succès ou d'erreur.

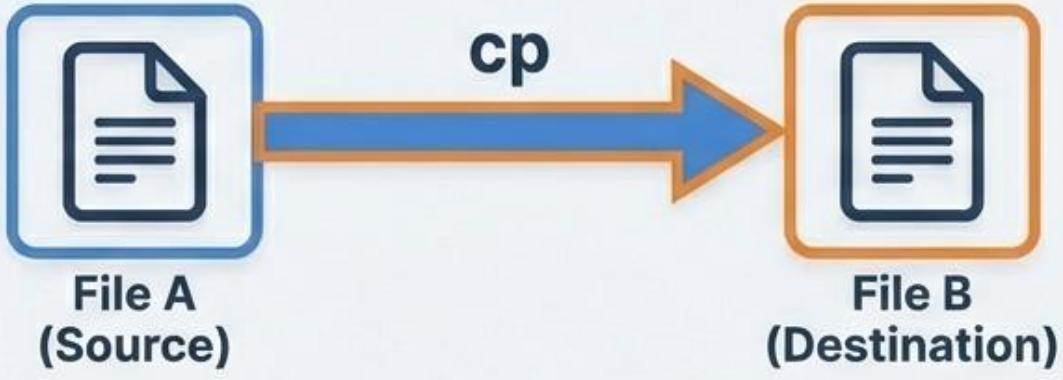


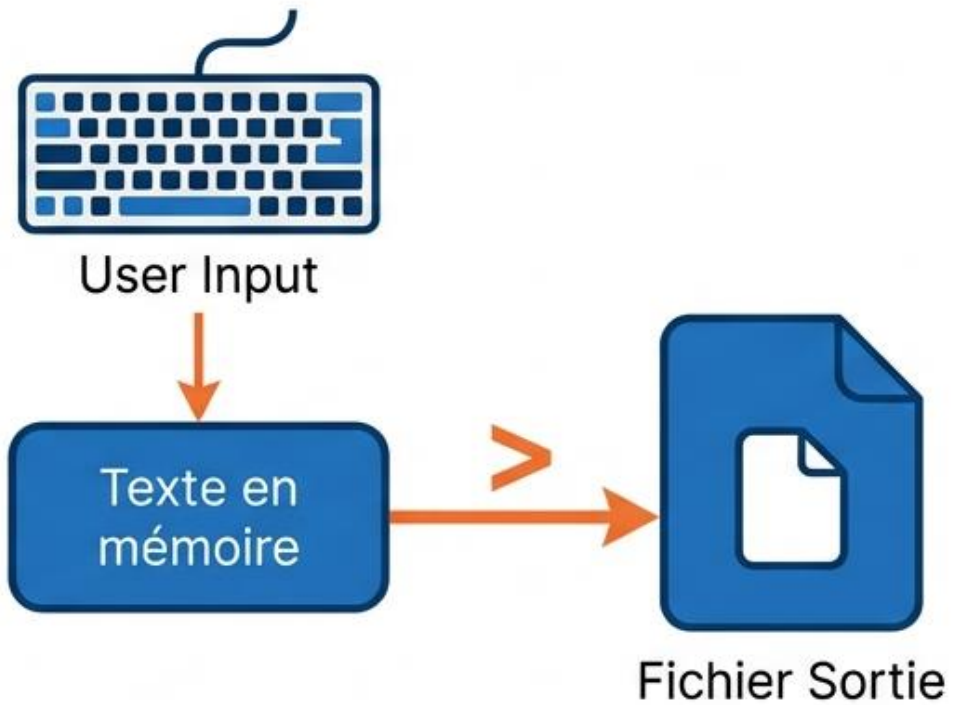
Schéma de la copie de fichier

Exercice 4 : Création et Redirection

Objectif : Créer un script qui enregistre du texte utilisateur dans un nouveau fichier.

Algorithme à implémenter :

1. **Capturer le contenu** : Utiliser `read` pour prendre une ligne de texte.
2. **Choisir la cible** : Demander le nom du fichier de sortie.
3. **Redirection** : Utiliser l'opérateur `>` pour écrire le texte saisi dans le fichier nommé.



Résumé des commandes

Interaction

`read` : Lecture entrée

`echo` : Écriture sortie

`echo -e` : Interprétation \n

Flux

`>` : Redirection (Écraser)

`>>` : Redirection (Ajout)

`|` : Pipe (Connexion)

Tests Fichiers

`-e` : Existe

`-f` : Fichier

`-d` : Dossier

`-r / -w / -x` : Permissions

Vos scripts sont désormais capables d'interagir avec l'utilisateur et le système de fichiers.