? Comprendre les redirections : stdout, stderr, 2>&1, /dev/null et tee

Les redirections sont l'un des concepts les plus importants à maîtriser sous Linux. Elles permettent de contrôler où sont envoyés les résultats d'une commande et les messages d'erreur.

? Sommaire

? Les flux standards Linux

> **Information**
> Chaque programme Linux utilise trois flux standards.

Flux	Numéro	Description
stdin	0	Entrée standard (clavier)
stdout	1	Sortie standard
stderr	2	Sortie d'erreur

Schéma général

```
             Clavier
                |
                v
            stdin (0)

             COMMANDE

stdout (1) -------------> Écran

stderr (2) -------------> Écran
```

Par défaut, les résultats et les erreurs sont affichés sur le terminal.

? Redirection de stdout (1)

Le symbole `>` redirige la sortie standard.

```
ls /etc > resultat.txt
```

Ce qui se passe

```
stdout (1)
    |
    v
resultat.txt
```

Le contenu n'est plus affiché à l'écran mais enregistré dans le fichier.

Écrasement vs Ajout

Syntaxe	Effet
`>`	Écrase le fichier
`>>`	Ajoute à la fin du fichier

Exemple

```
echo "Bonjour" > test.txt
echo "Monde" > test.txt
```

Résultat :

```
Monde
```

Le premier contenu a été remplacé.

?? Redirection de stderr (2)

Les erreurs utilisent un flux différent : stderr.

```
ls fichier_inexistant
```

Affiche :

```
ls: impossible d'accéder à 'fichier_inexistant'
```

Pour rediriger les erreurs :

```
ls fichier_inexistant 2> erreurs.log
```

Schéma

```
stderr (2)
    |
    v
erreurs.log
```

? Séparer stdout et stderr

```
ls /etc fichier_inexistant > sortie.log 2> erreur.log
```

Fichier	Contenu
sortie.log	Résultat normal
erreur.log	Messages d'erreur

? Comprendre 2>&1

> **Astuce importante**
> C'est probablement la syntaxe la plus utilisée dans les scripts Linux.

Commande

```
commande > fichier.log 2>&1
```

Décomposition

Cette partie :

```
> fichier.log
```

équivaut à :

```
1> fichier.log
```

Donc :

```
stdout --> fichier.log
```

Puis :

```
2>&1
```

signifie :

```
rediriger stderr vers le même emplacement que stdout
```

Schéma

```
stdout (1) -----\
                 \
                  ---> fichier.log
                 /
stderr (2) -----/
```

Résultat

```
commande > fichier.log 2>&1
```

Les sorties normales et les erreurs sont regroupées dans le même fichier.

?? Pourquoi l'ordre est important ?

Correct :

```
commande > fichier.log 2>&1
```

Incorrect :

```
commande 2>&1 > fichier.log
```

> **Attention**
> Linux traite les redirections de gauche à droite.

Dans le second cas, stderr continue d'utiliser le terminal.

? Syntaxe simplifiée : &>

Bash fournit un raccourci :

```
commande &> fichier.log
```

Équivalent à :

```
commande > fichier.log 2>&1
```

Ajout dans le fichier

```
commande &>> fichier.log
```

?? Comprendre /dev/null

> **Information**
> /dev/null est la poubelle de Linux.

Tout ce qui est envoyé dans ce fichier disparaît immédiatement.

```
echo "test" > /dev/null
```

Aucun résultat.

Ignorer les erreurs

```
commande 2>/dev/null
```

Ignorer la sortie normale

```
commande >/dev/null
```

Tout ignorer

```
commande >/dev/null 2>&1
```

Schéma

```
stdout --------> /dev/null

stderr --------> stdout
                    |
                    v
                /dev/null
```

Rien ne s'affiche.

? Comprendre la commande tee

La commande `tee` permet d'écrire à la fois :

- dans un fichier
- sur l'écran

Exemple

```
ls | tee resultat.log
```

Schéma

```
stdout
   |
   +------> écran
   |
   +------> resultat.log
```

L'utilisateur voit le résultat tout en conservant une trace dans un fichier.

? Tee avec sudo

Cette commande échoue :

```
sudo echo "8.8.8.8 google" > /etc/hosts
```

Car la redirection est exécutée par le shell utilisateur avant sudo.

Solution

```
echo "8.8.8.8 google" | sudo tee -a /etc/hosts
```

Schéma

```
echo
 |
 v
sudo tee -a
 |
 +------> écran
 |
 +------> /etc/hosts
```

Ici, c'est bien tee qui écrit dans le fichier avec les privilèges root.

? Résumé rapide

Syntaxe	Action
`commande > fichier`	stdout vers fichier
`commande >> fichier`	Ajouter stdout
`commande 2> fichier`	stderr vers fichier
`commande > fichier 2>&1`	stdout + stderr vers fichier
`commande &> fichier`	Version courte Bash
`commande >/dev/null`	Ignorer stdout
`commande 2>/dev/null`	Ignorer stderr

Syntaxe	Action
`commande >/dev/null 2>&1`	Tout ignorer
`commande | tee fichier`	Afficher + enregistrer
`commande | sudo tee -a fichier`	Afficher + enregistrer avec sudo

À retenir

Les 4 syntaxes les plus utilisées en administration Linux sont :

- `commande > fichier.log 2>&1`
- `commande >> fichier.log 2>&1`
- `commande >/dev/null 2>&1`
- `commande | tee fichier.log`

Revision #1
Created 2026-06-11 08:49:17 UTC by Nico là
Updated 2026-06-11 08:50:01 UTC by Nico là